CNCO2000 – Practical 07

P07: Networking with TCP

Q1:Understand the Basics of Transport Control Protocol

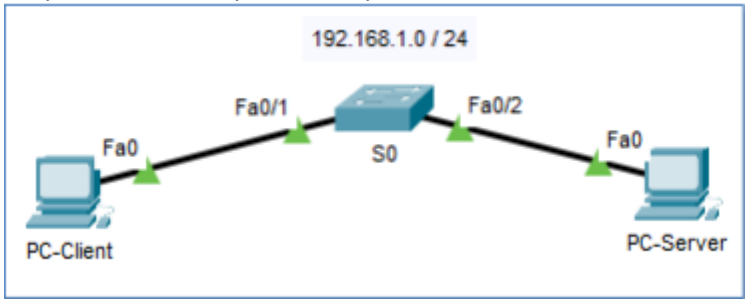
The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and errorchecked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP, which is part of the Transport Layer of the TCP/IP suite. SSL/TLS often runs on top of TCP.

TCP is connection-oriented, and a connection between client and server is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection is established. Three-way handshake (active open), retransmission, and error-detection adds to reliability but lengthens latency. TCP employs network congestion avoidance. However, there are vulnerabilities to TCP including denial of service, connection hijacking, TCP veto, and reset attack. For network security, monitoring, and debugging, TCP traffic can be intercepted and logged with a packet sniffer.

Though TCP is a complex protocol, its basic operation has not changed significantly since its first specification. TCP is still dominantly used for the web, i.e. for the HTTP protocol, and later HTTP/2, while not used by latest standard HTTP/3.

Q2: Configuring a TCP Client and a TCP Server

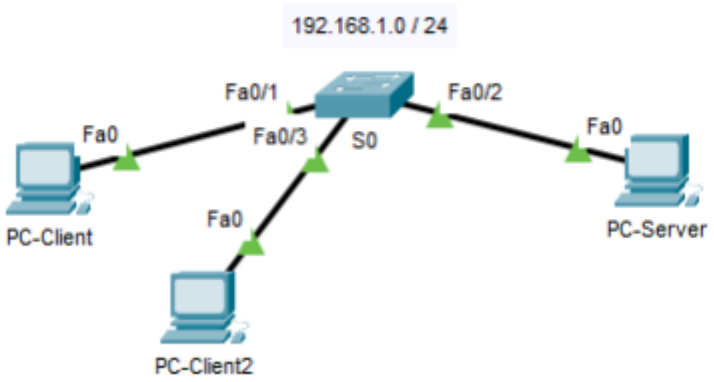
• Open PTLab 07.2.pka and implement the network shown below:



In this activity, a TCP socket in Python will be used to send data from PC-Client to PC-Server.

Q3: Try me! Questions

1. Connect another client to S0 as shown below and forward messages from PC-Client to PC-Client2 (via PCServer) vice versa. (you may assume PC-Client, PC-Client2 are two chat clients)

2. 

2. In Q2, try to simulate a packet drop during the data transmission from PC-Client to PC-Server. See whether TCP recovers from the lost packet.

LAB 07.2

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC-Client
PT-PC	PC-Server
2960	S0

b) Connect them using copper-cables according to the table below:

Device Name	Interface	Device Name	Interface
PC-Client	Fa0	S0	Fa0/1
PC-Server	Fa0	S0	Fa0/2

2. Configuring devices

a) PCs need to be given IP addresses (static IP assignment) as shown below:

PC Name	Interface	IP Address

PC-Client	Fa0	192.168.1.2/255.255.255.0
PC-Server	Fa0	192.168.1.3/255.255.255.0

b) Configure the TCP Server (PC-Server):

a. Go to PC-Server -> Programming -> New

b. Name: **TCP Server**, Template: **TCP Server – Python**

c. Click on main.py

```
# Importing the libraries required TCP socket communication
from tcp import *
from time import *

port = 1234 # Server will be listening to new connections on this port
server = TCPServer()

def onTCPNewClient(client):

   # Upon the connection state change, this method will be called
   # For TCP state diagram, please refer to P07 reference material on
   # Blackboard
   def onTCPConnectionChange(type):

      # Print the remote client IP on the console
      print("connection to " + client.remoteIP() + " changed to state " +
      str(type))
   # Upon receiving a data from client, this method will be called
   def onTCPReceive(data):
print("received from " + client.remoteIP() + " with data: " + data)

# send back same data (echo data back to the client)
client.send(data)

   # Register the callbacks above on the client object, so that when there
   is # a connection state change, or/and data is received from a remote
   client, # the methodsonTCPConnectionChange() and onTCPReceive() is
   called
   # respectively.
   client.onConnectionChange(onTCPConnectionChange)
   client.onReceive(onTCPReceive)

# Main method of the client
def main():

     # Call onTCPNewClient() method upon receiving a new connection via
     the # listening port defined above (e.g. 1234)
server.onNewClient(onTCPNewClient)
# start listening on the port defined (e.g. 1234) and print the status
# this is a blocking call
     print(server.listen(port))
```

```
# don’t let it finish, enter into the main program loop
while True:
sleep(3600)

# Invoke the main method
if __name__ == "__main__":
main()
```

c) Configure the TCP Client (PC-Client):

a. Go to PC-Server -> Programming -> New

b. Name: **TCP Client**, Template: **TCP Client – Python**

c. Click on main.py and change **serverIP, serverPort** variables as shown below:

```
# Importing the libraries required TCP socket communication
from tcp import *
from time import *

serverIP = "192.168.1.3" # Server IP to connect
serverPort = 1234 # Server port to connect

client = TCPClient()

# Upon the connection state change, this method will be called
# For TCP state diagram, please refer to P07 reference material on
# Blackboard
def onTCPConnectionChange(type):
     print("connection to " + client.remoteIP() + " changed to state " +
     str(type))

# Upon receiving a data from server, this method will be called
def onTCPReceive(data):
print("received from " + client.remoteIP() + " with data: " + data)


# Main method of the client
def main():
     # Register the callbacks above on the client object (client object
     # represents the server here), so that when there is a connection
     state # change, or/and data is received from a remote client, the
     methods
     # onTCPConnectionChange() and onTCPReceive() is called respectively.
client.onConnectionChange(onTCPConnectionChange)
client.onReceive(onTCPReceive)

# initiate connection to the server and print the status
print(client.connect(serverIP, serverPort))

# Send data “hello <count>” every 5 seconds
# Note that TCP is a byte stream protocol. The string “hello <count>”
     # will be sent as a stream of bytes in TCP
```

```
count = 0
while True:
count += 1
data = "hello " + str(count)
client.send(data) # send as a stream of bytes
sleep(5)

# Invoke the main method
if __name__ == "__main__":
main()
```

3. Validating the connection

a) Open PC-Server -> TCP Server *(Software Project)* -> main.py -> Click on **run**

b) Open PC-Client -> TCP Client *(Software Project)* -> main.py -> Click on **run.** stop the program after sending 10 messages.

c) Observe the following console prints at the PC-Server

```
Starting TCP Server (Python)...
True // listening port status print
received from 192.168.1.2 with data: hello 2 // print data from client
received from 192.168.1.2 with data: hello 3 // print data from client
received from 192.168.1.2 with data: hello 4 // print data from client
received from 192.168.1.2 with data: hello 5 // print data from client
received from 192.168.1.2 with data: hello 6 // print data from client
received from 192.168.1.2 with data: hello 7 // print data from client
received from 192.168.1.2 with data: hello 8 // print data from client
received from 192.168.1.2 with data: hello 9 // print data from client
received from 192.168.1.2 with data: hello 10 // print data from client
connection to 192.168.1.2 changed to state 3 // connection state
                                             // (connection-terminated)
```

d) Observe similar console prints at PC-Client

```
Starting TCP Client (Python)...
True // status of initiating the connection with the server
connection to 192.168.1.3 changed to state 0 // connection state
                                             // (connection-established)
received from 192.168.1.3 with data: hello 2 // print data from server
received from 192.168.1.3 with data: hello 3 // print data from server
received from 192.168.1.3 with data: hello 4 // print data from server
received from 192.168.1.3 with data: hello 5 // print data from server
received from 192.168.1.3 with data: hello 6 // print data from server
received from 192.168.1.3 with data: hello 7 // print data from server
```

```
received from 192.168.1.3 with data: hello 8 // print data from server
received from 192.168.1.3 with data: hello 9 // print data from server
received from 192.168.1.3 with data: hello 10 // print data from server
TCP Client (Python) stopped. // This will initiate connection termination
```

4. Investigating TCP traffic

a) While validating the connection according to the previous step, switch to simulation mode and observe the TCP packets (TCP header, TCP segment data – byte stream, IP header, etc.)

CNCO2000 – Practical 08

P08: Networking with UDP Q1:Understand the Basics of User Datagram Protocol

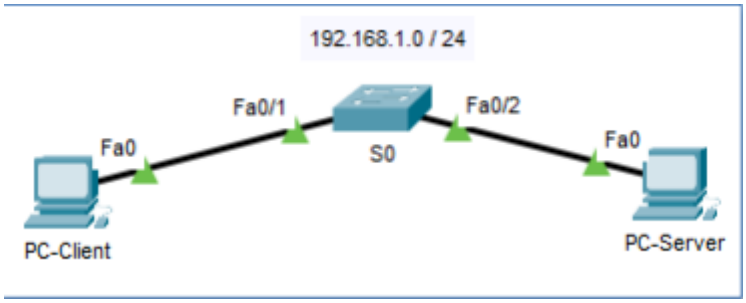
In computer networking, the User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768. With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network. Prior communications are not required in order to set up communication channels or data paths.

UDP uses a simple connectionless communication model with a minimum of protocol mechanisms. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection. If error-correction facilities are needed at the network interface level, an application may use Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.

UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing in the protocol stack. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system.

Q2: Configuring a UDP Client and a UDP Server

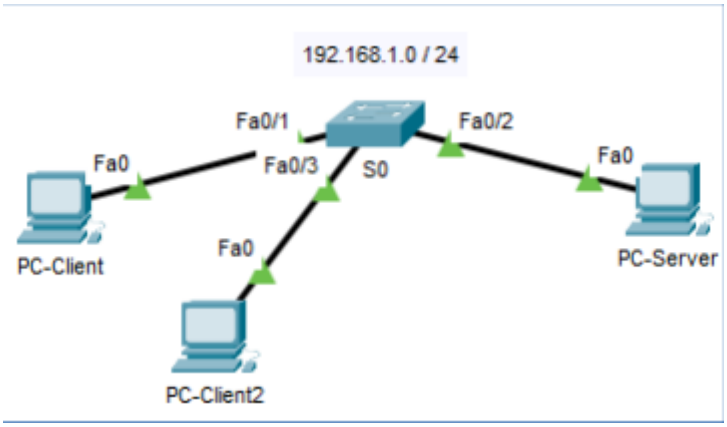
• Open PTLab 08.2.pka and implement the network shown below:



In this activity, a TCP socket in Python will be used to send data from PC-Client to PC-Server.

Page Q3: Try me! Questions

1. Connect another client to S0 as shown below and forward messages from PC-Client to PC-Client2 (via PCServer) vice versa. (you may assume PC-Client, PC-Client2 are two chat clients)



2. In Q2, try to simulate a packet drop during the data transmission from PC-Client to PC-Server. See whether UDP recovers from the lost packet.

LAB 08.2

Note that in UDP there is no connection establishment or termination phase. UDP is a connectionless protocol unlike TCP. In this example, both the client and the server create UDP socket to send and receive data.

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC-Client
PT-PC	PC-Server
2960	S0

b) Connect them using copper-cables according to the table below:

Device Name	Interface	Device Name	Interface
PC-Client	Fa0	S0	Fa0/1
PC-Server	Fa0	S0	Fa0/2

2. Configuring devices

a) PCs need to be given IP addresses (static IP assignment) as shown below:

PC Name	Interface	IP Address

PC-Client	Fa0	192.168.1.3/255.255.255.0
PC-Server	Fa0	192.168.1.2/255.255.255.0

b) Configure the UDP Server (PC-Server):

a. Go to PC-Server -> Programming -> New

b. Name: **UDP Server**, Template: **UDP Socket – Python**

c. Click on main.py

```
# Importing the libraries required UDP socket communication
from udp import *
from time import *

# Upon receiving a data from client, this method will be called
def onUDPReceive(ip, port, data):
print("received from " + ip + ":" + str(port) + ":" + data);

# Main method of the client
def main():
socket = UDPSocket()

     # Call onUDPReceive() method upon receiving data - optional
socket.onReceive(onUDPReceive)

# open a socket on this PC to receive UDP data on port 1235
print(socket.begin(1235))

count = 0
while True:
count += 1
            # Note that there is no connection establishment in UDP.
            # We directly send data to the other side.
            # Destination Socket = 192.168.1.2:1235
socket.send("192.168.1.2", 1235, "hello " + str(count))
sleep(5)

# Invoke the main method
if __name__ == "__main__":
main()
```

c) Configure the UDP Client (PC-Client):

a. Go to PC-Server -> Programming -> New

b. Name: **UDP Client**, Template: **UDP Socket – Python**

c. Click on main.py and change **serverIP, serverPort** variables as shown below:

```
# Importing the libraries required UDP socket communication
from udp import *
from time import *

# Upon receiving a data from server, this method will be called
def onUDPReceive(ip, port, data):
print("received from " + ip + ":" + str(port) + ":" + data);

# Main method of the client
def main():
socket = UDPSocket()

     # Call onUDPReceive() method upon receiving data - optional
socket.onReceive(onUDPReceive)

# open a socket on this PC to receive UDP data on port 1235
print(socket.begin(1235))

count = 0
while True:
count += 1
            # Note that there is no connection establishment in UDP.
            # We directly send data to the other side.
            # Destination Socket = 192.168.1.3:1235
            # The string “hello <count>” will be sent as a message but not
            as # a stream of bytes like in TCP. UDP preserves message
            # boundaries. If the message is lost, there will be no
            recovery

socket.send("192.168.1.3", 1235, "hello " + str(count))
sleep(5)

# Invoke the main method
if __name__ == "__main__":
main()
```

3. Validating the connection

a) Open PC-Server -> UDP Server *(Software Project)* -> main.py -> Click on **run**

b) Open PC-Client -> UDP Client *(Software Project)* -> main.py -> Click on **run.** stop the program after sending 9 messages.

c) Observe the following console prints at the PC-Server

```
Starting UDP Server (Python)...
True // open socket status on port 1235
received from 192.168.1.2:1235:hello 1 // print data from client
received from 192.168.1.2:1235:hello 2 // print data from client
```

```
received from 192.168.1.2:1235:hello 3 // print data from client
received from 192.168.1.2:1235:hello 4 // print data from client
received from 192.168.1.2:1235:hello 5 // print data from client
received from 192.168.1.2:1235:hello 6 // print data from client
received from 192.168.1.2:1235:hello 7 // print data from client
received from 192.168.1.2:1235:hello 8 // print data from client
received from 192.168.1.2:1235:hello 10 // print data from client
UDP Server (Python) stopped.
```

d) Observe similar console prints at PC-Client

```
Starting UDP Client (Python)...
True // open socket status on port 1235
received from 192.168.1.3:1235:hello 2 // print data from server
received from 192.168.1.3:1235:hello 3 // print data from server
received from 192.168.1.3:1235:hello 4 // print data from server
received from 192.168.1.3:1235:hello 5 // print data from server
received from 192.168.1.3:1235:hello 7 // print data from server
received from 192.168.1.3:1235:hello 8 // print data from server
received from 192.168.1.3:1235:hello 9 // print data from server
received from 192.168.1.3:1235:hello 10 // print data from server
UDP Client (Python) stopped.
```

4. Investigating UDP traffic

a) While validating the connection according to the previous step, switch to simulation mode and observe the UDP packets (UDP header, UDP message data – message stream, IP header, etc.)

b) To simulate UDP packet drop, turn off the port Fa0/1 or Fa0/2 on the switch for a few seconds during the exchange of UDP packets between PC-Client and PC-Server. Observe the console prints at both sides (you should see a sequence gap on hello messages)

CNCO2000 – Practical 09

P09: Networking with Application Layer I (Telnet, FTP, HTTP, Email)

Q1:Understand the Basics of Application Layer Protocols

An application layer is an abstraction layer that specifies the shared communications protocols and interface methods used by hosts in a communications network. The application layer abstraction is used in both of the standard models of computer networking: The Internet Protocol Suite (TCP/IP) and the OSI model. Although both models use the same term for their respective highest-level layer, the detailed definitions and purposes are different.

Telnet

Telnet stands for the TELecomunications NETwork. It helps in terminal emulation. It allows Telnet client to access the resources of the Telnet server. It is used for managing the files on the internet. It is used for initial set up of devices like switches. The telnet command is a command that uses the Telnet protocol to communicate with a remote device or system. Port number of telnet is 23.

FTP

FTP stands for file transfer protocol. It is the protocol that actually lets us transfer files.It can facilitate this between any two machines using it. But FTP is not just a protocol but it is also a program.FTP promotes sharing of files via remote computers with reliable and efficient data transfer. Port number for FTP is 20 for data and 21 for control.

TFTP

The Trivial File Transfer Protocol (TFTP) is the stripped-down, stock version of FTP, but it's the protocol of choice if you know exactly what you want and where to find it. It's a technology for transferring files between network devices and is a simplified version of FTP

HTTP

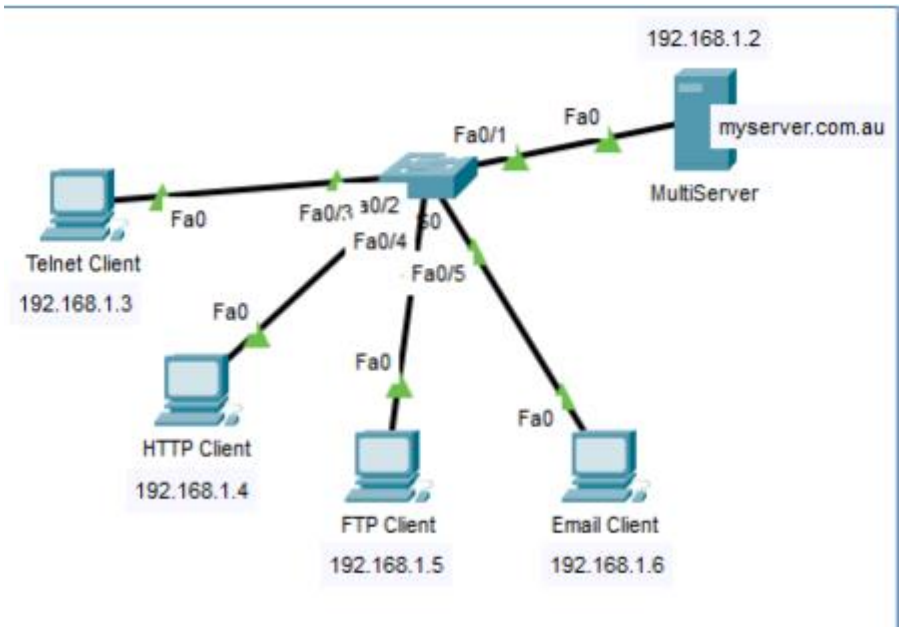
The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser.

SMTP

It stands for Simple Mail Transfer Protocol. It is a part of the TCP/IP protocol. Using a process called "store and forward," SMTP moves your email on and across networks. It works closely with something called the Mail Transfer Agent (MTA) to send your communication to the right computer and email inbox. Port number for SMTP is 25.

Q2: Configuring a MultiServer Environment for Telnet, FTP, HTTP, Email

• Open PTLab 09.2.pka and implement the network shown below:



Q3: Try me! Questions

1. Register another email account for the user (euser2) at MultiServer and connect another Email Client (PC2- Email).

a. Configure euser2@myserver.com.au account on the newly connected PC (PC2-Email).

b. Send an email from euser@myserver.com.au to euser2@myserver.com.au.

c. Check whether PC2-Email has received the email above.

d. Reply to the email received above and check whether the reply is received by PC (Email Client)

LAB 09.2

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-Server	MultiServer
PT-PC	Telnet Client
PT-PC	HTTP Client
PT-PC	FTP Client
PT-PC	Email Client
2960	S0

b) Connect them using copper-cables according to the table below:

Device Name	Interface	Device Name	Interface

MultiServer	Fa0	S0	Fa0/1
Telnet Client	Fa0	S0	Fa0/2
HTTP Client	Fa0	S0	Fa0/3
FTP Client	Fa0	S0	Fa0/4
Email Client	Fa0	S0	Fa0/5

2. Configuring devices

a) Devices need to be given IP addresses (static IP assignment) as shown below:

DeviceName	Interface	IP Address
MultiServer	Fa0	192.168.1.2/255.255.255.0
Telnet Client	Fa0	192.168.1.3/255.255.255.0
HTTP Client	Fa0	192.168.1.4/255.255.255.0
FTP Client	Fa0	192.168.1.5/255.255.255.0
Email Client	Fa0	192.168.1.6/255.255.255.0
S0	vlan 1	192.168.1.10/255.255.255.0

3. Configuring DNS

// DNS Server configurations will be explained in detail in the next practical sheet (P10 Networking with Application Layer II). DNS is used to resolve the ASCII (easy to remember) domain name to an IP address.

a) Go to MultiServer -> Service -> DNS and add the following "**A Record**"s

Name: myserver.com.au, **Address:** 192.168.1.2, **Type:** A Record

Name: mail.myserver.com.au, **Address:** 192.168.1.2, **Type:** A Record

b) Configure DNS Server IP on following PCs:

PC Name	DNS Server
Telnet Client	192.168.1.2
HTTP Client	192.168.1.2
FTP Client	192.168.1.2
Email Client	192.168.1.2

Hint: PC -> Desktop -> IP Configuration

4. Configure/Validate Telnet

a) Set the password **"cisco"** on line vty 0 of the Switch **S0**

b) Use telnet to connect to the Swtich **S0** from **Telnet Client**

c) Swtich to simulation mode and observe the Telnet traffic

d) Observe the underlying Transport Layer Protocol for Telnet

5. Configure/Validate HTTP

a) On **MultiServer**, Go to Services -> **HTTP**, and turn it on.

b) On **HTTP Client**, Go to Desktop-> **Web Browser** and type **192.168.1.2**

   a. Check whether you see a website from the MultiServer

   b. Repeat step a by typing "myserver.com.au" // In here, the DNS will resolve the domain name to IP address for you.

c) Swtich to simulation mode and observe the HTTP traffic

d) Observe the underlying Transport Layer Protocol for HTTP

6. Configure/Validate FTP

a) On **MultiServer**, Go to Services -> **FTP**, and turn it on.

   a. Create a new user account with write, read, delete, rename, list privileges

   b. **Username:** cc_user, **Password:** cc_pass

b) On **MultiServer**, Go to Desktop -> **Text Editor**, and type "Hello World" on a new file called "text_on_server.txt"

   a. Hint: type the text and close the Text Editor, it will prompt you to save.

c) On **FTP Client**, Go to Desktop-> **Command Prompt**, and type:

   a. `ftp 192.168.1.2 // or ftp myserver.com.au`

   b. Use the user credentials created above to login to FTP Server at MultiServer

   c. `ftp> ? // ? will list the ftp commands that can be used on FTP Server`

   d. Fetch "text_on_server.txt" file from the FTP Server at MultiServer

      (Hint: you may use get, dir, cd commands)

   e. Delete "text_on_server.txt" from the FTP Server at MultiServer

   f. ftp> quit `// exit from ftp prompt. Terminate the FTP connection`

e) Swtich to simulation mode and observe the FTP traffic
f) Observe the underlying Transport Layer Protocol for FTP

Some useful tools/commands to manipulate files on a PC

PC -> Desktop -> TFTP Service (TFTP: Trivial FTP) to view the files

PC -> Desktop -> Command Prompt, and use the commands

- cd, dir, delete, mkdir

7. Configure/Validate Email

a) On **MultiServer**, Go to Services -> **EMAIL**, and
   a. Turn on SMTP and POP3
   b. Set the Mail Exchange Domain Name: myserver.com.au
   c. Add a user with (Username: euser, Password: epass)
b) On **Email Client**, Go to Desktop-> **Email** -> Configure Mail
   a. **Your Name:** Electronic User // This is not the username but your display name
   b. **Email Address:** euser@myserver.com.au
   c. **Incoming mail server:** mail.myserver.com.au // or 192.168.1.2, Note that "mail.myserver.com.au" must resolve to 192.168.1.2
   d. **Outgoing mail server:** mail.myserver.com.au
   e. **User:** euser
   f. **Password:** epass
c) On **Email Client**, Go to Desktop-> **Email** and compose a test email to "euser@myserver.com.au" (here we are sending an email from euser to itself). Click on receive button to receive the email successfully.
d) Swtich to simulation mode and observe both SMTP, POP3 traffic
e) Observe the underlying Transport Layer Protocol for both SMTP, POP3

CNCO2000 – Practical 10

P10: Networking with Application Layer II (DHCP, DNS)

Q1:Understand the Basics of DHCP and DNS An application layer is an abstraction layer that specifies the shared communications protocols and interface methods used by hosts in a communications network. The application layer abstraction is used in both of the standard models of computer networking: The Internet Protocol Suite (TCP/IP) and the OSI model. Although both models use the same term for their respective highest-level layer, the detailed definitions and purposes are different.

DHCP

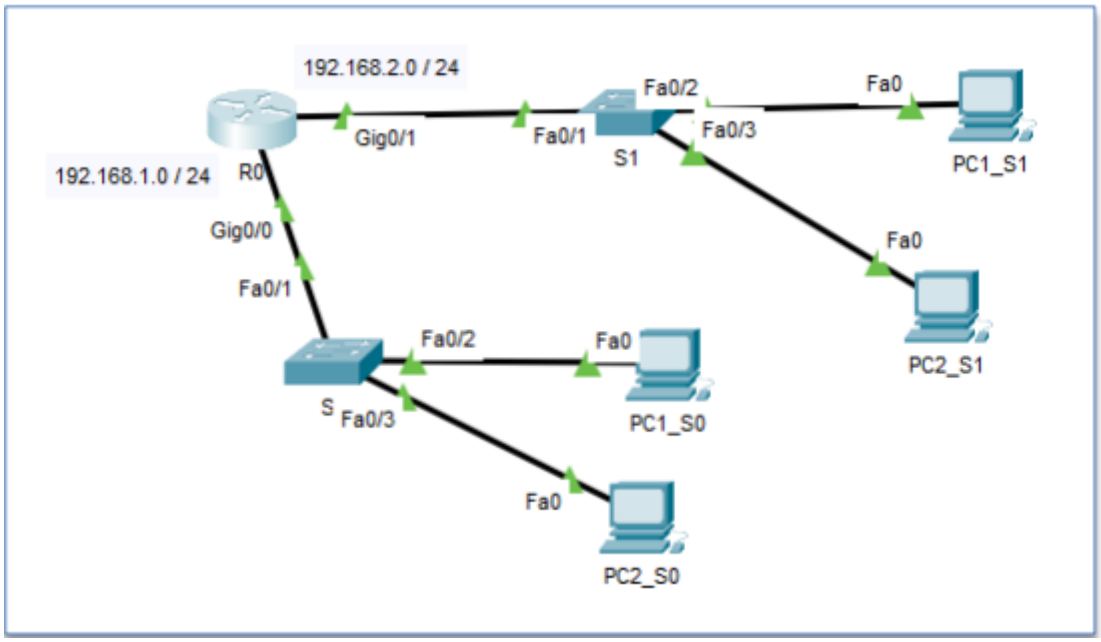
It stands for Dynamic Host Configuration Protocol (DHCP). It gives IP addresses to hosts. There is a lot of information a DHCP server can provide to a host when the host is registering for an IP address with the DHCP server. Port number for DHCP is 67, 68.

DNS

It stands for Domain Name System. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name www.abc.com might translate to 198.105.232.4. Port number for DNS is 53.

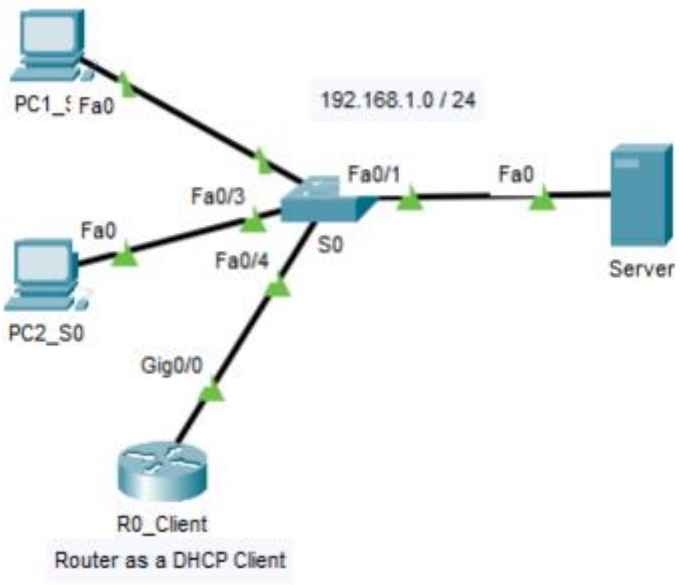
Q2: Configuring a DHCP Service on a Router

• Open PTLab 10.2.pka and implement the network shown below:



Q3: Configuring a DHCP Service on a Server PC

• Create a new .pkt file and implement the network shown below:



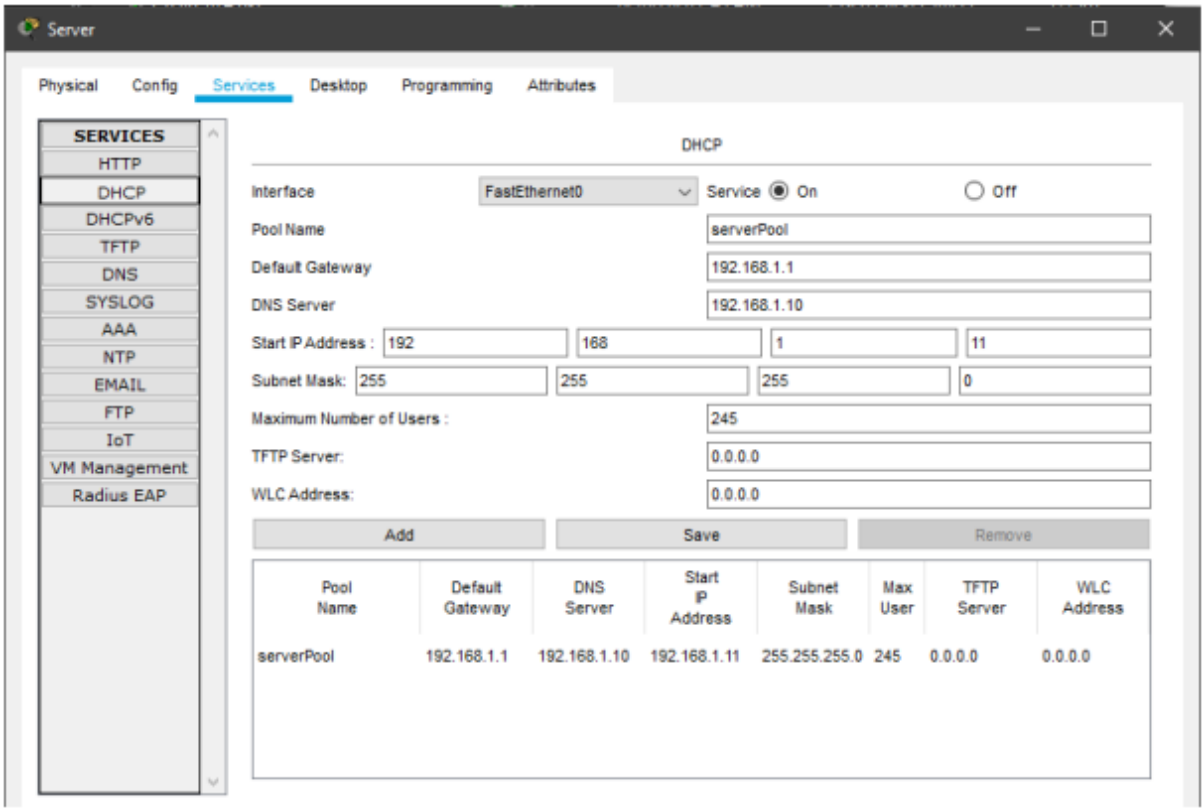
• Note: PCs and R0_Client are configured as DHCP Clients

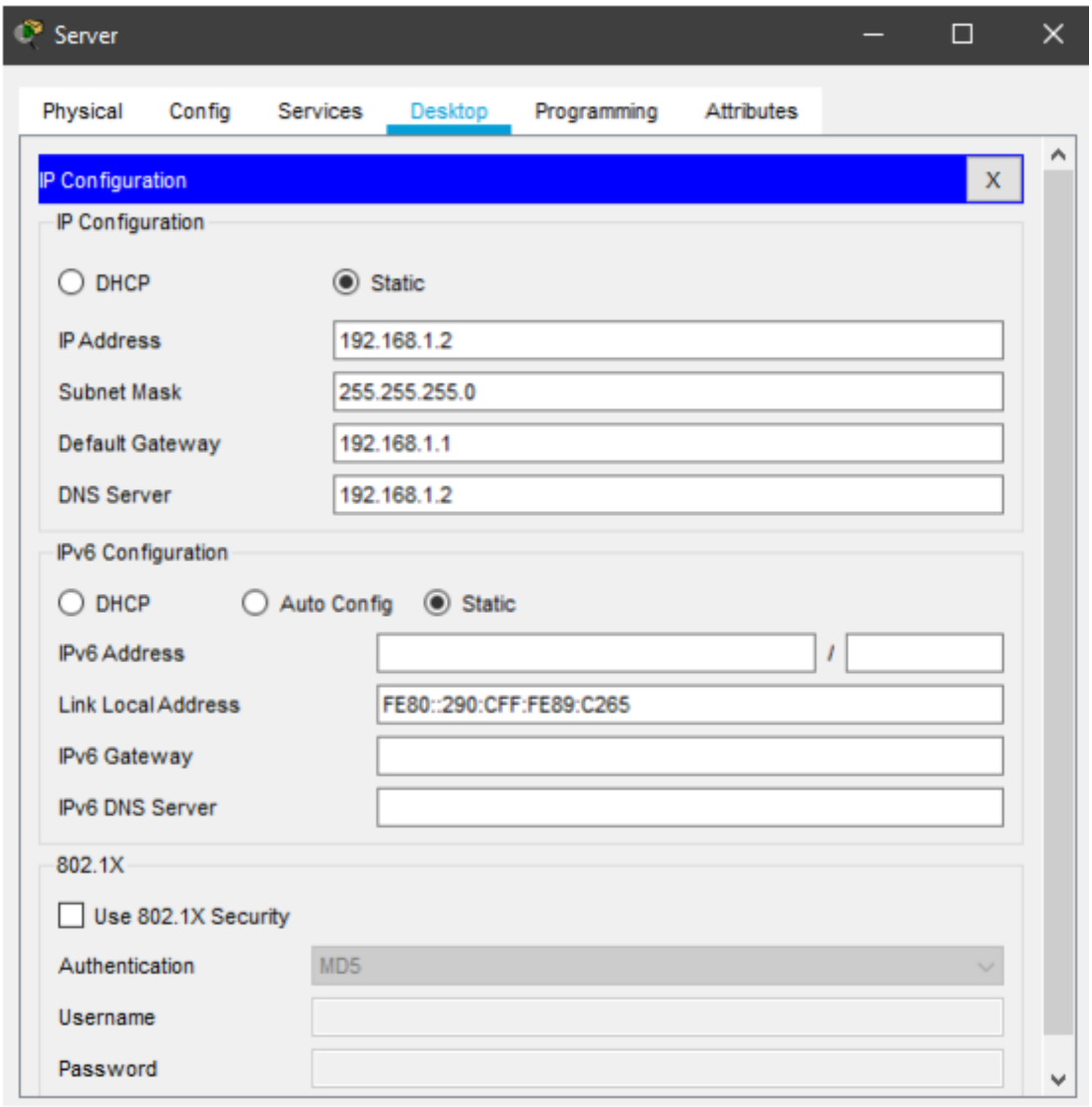
• To configure the Router as a DHCP client

R0_Client(config)# int Gig0/0

R0_Client(config-if)# ip address dhcp

• Configure the (DHCP) Server as shown below:



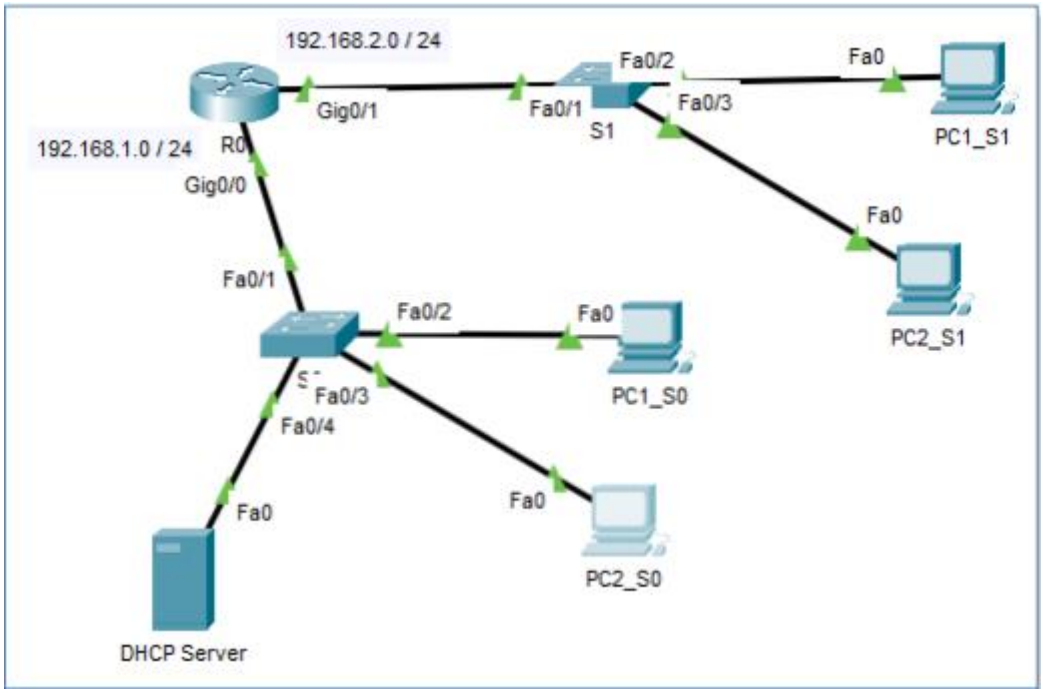


• Send an ICMP message from PC1_S0 to R0_Client, PC2_S0 client (must be successful)

• What is the critical problem in this configuration where a router is configured to obtain the default gateway address from a DHCP Server.

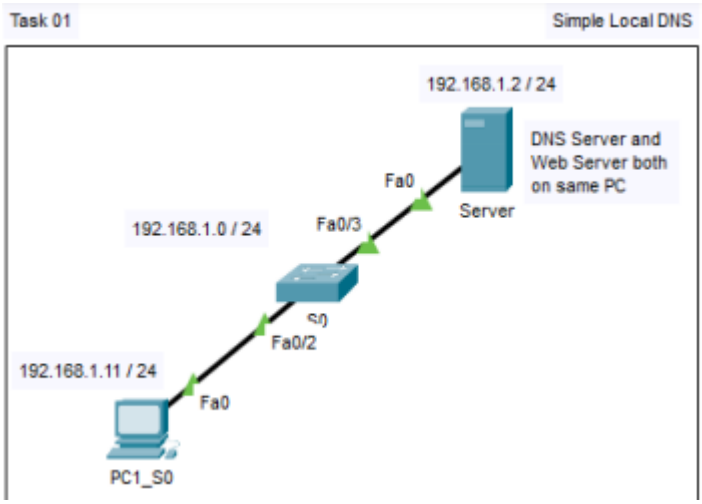
Q4: Configuring a DHCP Relay Agent

• Open PTLab 10.4.pka and implement the network shown below:

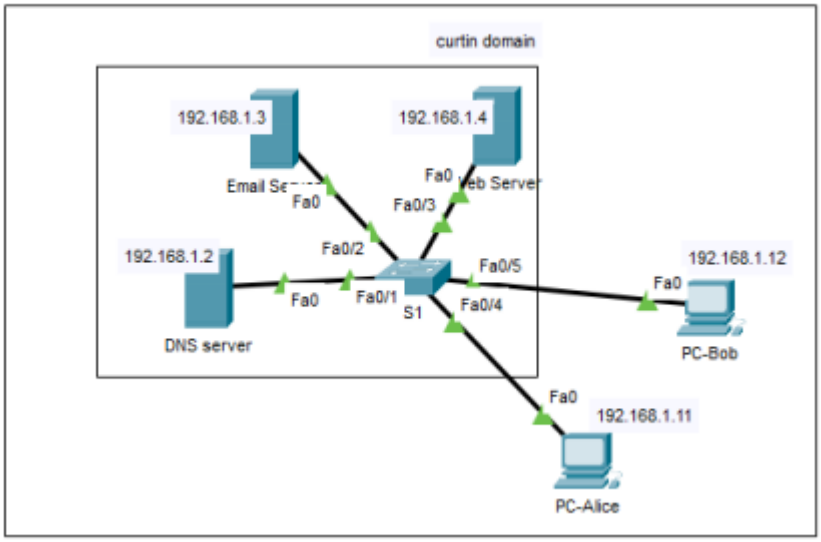


Q5: Configuring a Private/Local DNS Server (Single Level)

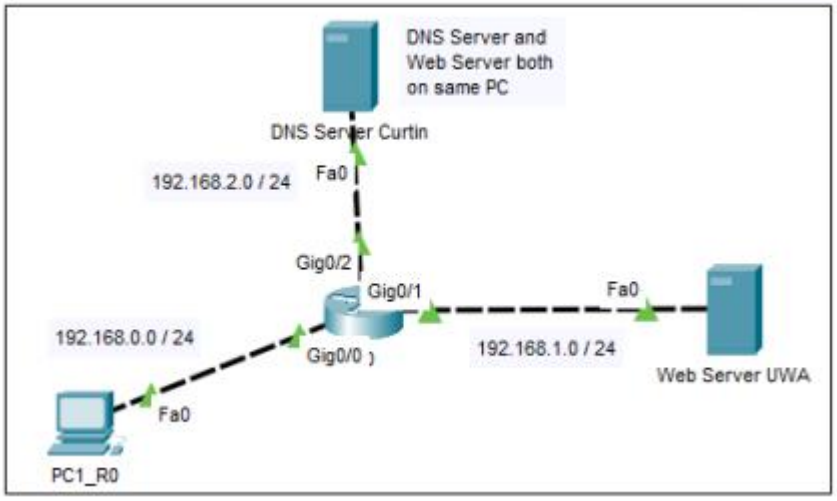
• Open PTLab 10.5.pka and implement the networks shown below:



Task 02

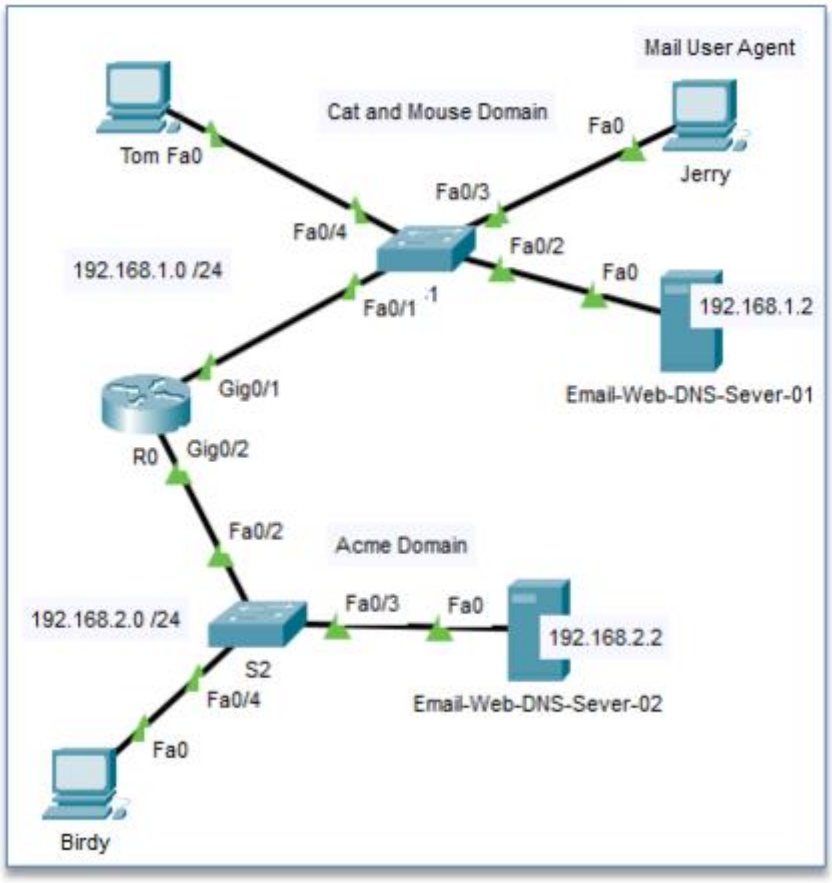


Task 03



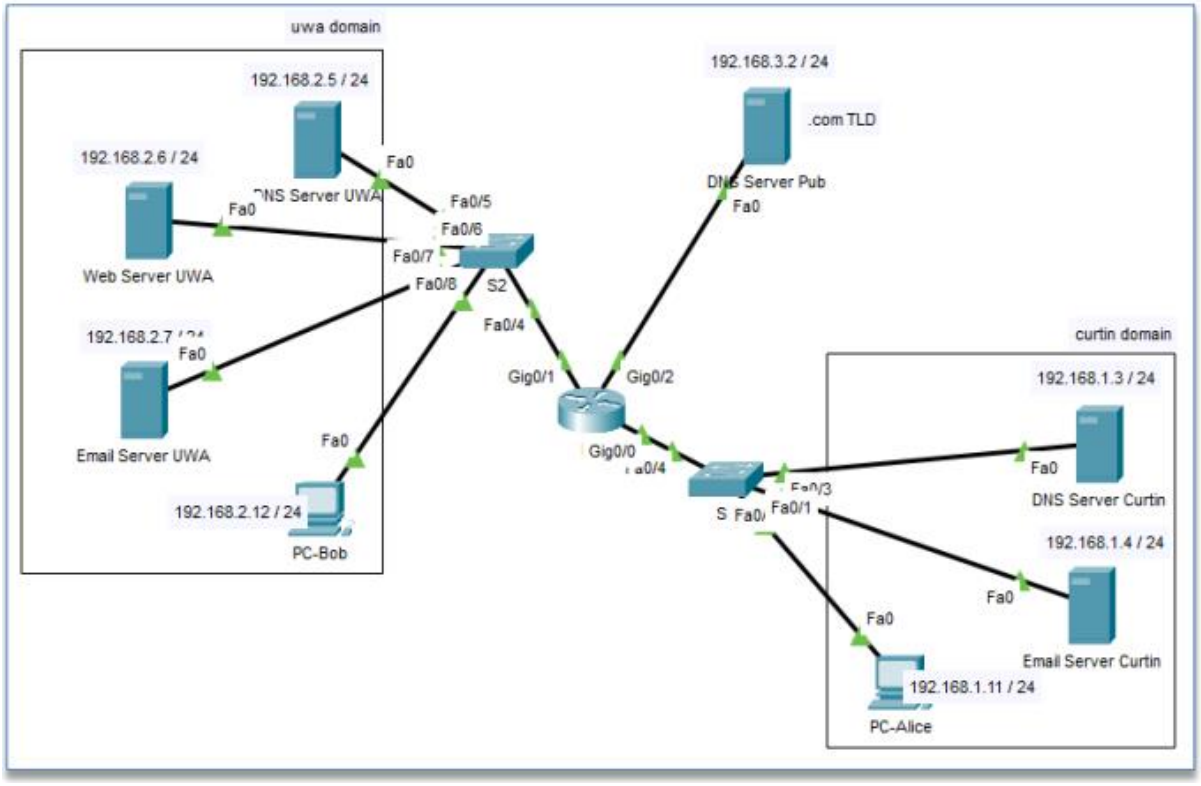
Q6: Configuring Two Local DNS/Email Servers

• Open PTLab 10.6.pka and implement the network shown below:



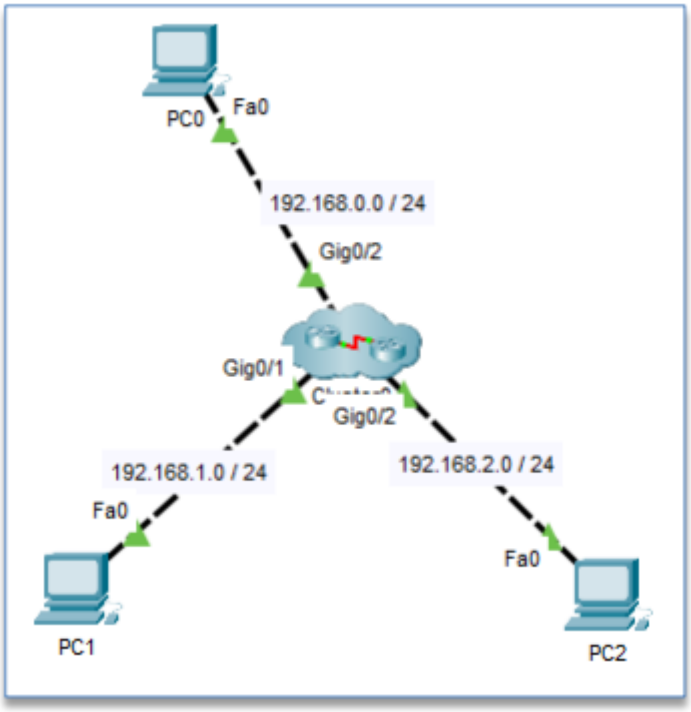
Q7: Configuring Multi-level DNS Servers

• Open PTLab 10.7.pka and implement the network shown below:



Q8: Configuring a p2p Network

• Open PTLab 10.8.pka and implement the network shown below:



Q7: Try me! Questions

1. In Q4, what would happen if R0 is also configured as a DHCP Server (additionally to the DHCP Server PC)? Will the network be stable?

2. In Q4, how does the DHCP Server knows which pool to be used when a DHCP Discover message is received from PC1_S0 and PC1_S1

3. In Q4, Configure two VLANs (Student- VLANID:20 and Staff- VLANID:30) on Swtich S1.

a. Assign PC1_S1, PC2_S2 to Student VLAN and Staff VLAN

b. Perform necessary configurations on the router so that the PCs on VLANs could obtain IP addresses from the existing DHCP Server

4. In Q7, what is the significance of having a SOA record at DNS Server UWA and another SOA record at DNS Server Pub? What would happen if the SOA record is removed at DNS Server Pub?

5. In Q7, it is explained why Alice's email to bob@uwa.com did not reach bob's inbox at Email Server UWA. On the same ground, explain why Alice's email (from alice@curtin.com) to bob@curtin.com reach bob's inbox in Q4? (Hint: Swtich to simulation mode and observer the DNS resolution process in sending the email)

6. In Q7, add necessary DNS entries so that bob@uwa.com can send an email to alice@curtin.com successfully.

7. In Q7, do necessary configuration changes in order to change .com domains to .edu.au domains (e.g. curtin.com => curtin.edu.au, uwa.com => uwa.edu.au, bob@uwa.com => bob@uwa.com.au,

etc.). Note that .edu and .au domains are same level TLDs (Hint: you may use two TLD servers, one for .au, another for .edu)

LAB 10.2

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC1_S0
PT-PC	PC2_S0
PT-PC	PC1_S1
PT-PC	PC2_S1
2960	S0
2960	S1
2911	R0

b) Connect them using copper-cables according to the table below:

(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC1_S0	Fa0	S0	Fa0/2
PC2_S0	Fa0	S0	Fa0/3
PC1_S1	Fa0	S1	Fa0/2
PC2_S1	Fa0	S1	Fa0/3
S0	Fa0/1	R0	Gig0/0
S1	Fa0/1	R0	Gig0/1

2. Configuring devices

a) Configure the default gateways according to the following table

Device Name	Interface	IP Address
R0	Gig0/0	192.168.1.1/255.255.255.0
R0	Gig0/1	192.168.2.1/255.255.255.0

(**Note:** make sure to turn on the ports of the router since the default is off)

b) Go to each PC -> Desktop -> IP Configuration and Click on **DHCP**to obtain an IP from the DHCP Service running on the Router.

c) **Configure DHCP Service on the Router**

```
// Define a DHCP pool of IP addresses to be assigned to hosts, a Default gateway for the LAN and a DNS Server.

// Can define multiple DHCP pools for multiple LANs

R0(config)# ip dhcp pool MY_LAN1 // DHCP Pool Name

R0(dhcp-config)# network 192.168.1.0 255.255.255.0 //Define a pool of IP address for the network 192.168.1.0/24

R0(dhcp-config)# default-router 192.168.1.1 // Default gateway for PCs

R0(dhcp-config)# dns-server 192.168.1.10 // DNS Server for PCs


// We can add ip dhcp excluded-address command to our configuration so as to configure the router to exclude addresses 192.168.1.1 through 192.168.1.10 when assigning addresses to clients. The ip dhcp excluded-address command may be used to reserve addresses that are statically assigned to key hosts.

R0(config)# ip dhcp excluded-address 192.168.1.1 192.168.1.10


// Configure a Pool (MY_LAN2) for the LAN 192.168.2.0/24 Network similarly

…

…


// Some useful commands to review the configuration

R0# show ip dhcp [pool|binding|conflict]

C:/>ipconfig [/release | /renew]
```

d) Once the configuration is complete, check on the IP addresses of PCs that are obtained from the DHCP service. (Note: Allocated IP addressees will start from 192.168.1.11 and 192.168.2.11 due to the exclusion of 192.168.1.1-10 and 192.168.2.1-10)

3. Validating the connection

a) Swtich to simulation mode and observe the DHCP handshake (DORA Process – Discover, Offer, Request, Acknowledge)

b) Send an ICMP message from PC1_S0 to PC1_S0

c) Send an ICMP message from PC1_S0 to PC2_S1

LAB 10.4

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC1_S0
PT-PC	PC2_S0
PT-PC	PC1_S1
PT-PC	PC2_S1
2960	S0
2960	S1
2911	R0
PT-Server	DHCP Server

b) Connect them using copper-cables according to the table below:

(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC1_S0	Fa0	S0	Fa0/2
PC2_S0	Fa0	S0	Fa0/3
PC1_S1	Fa0	S1	Fa0/2
PC2_S1	Fa0	S1	Fa0/3
S0	Fa0/1	R0	Gig0/0
S1	Fa0/1	R0	Gig0/1
DHCP Server	Fa0	S0	Fa0/4

2. Configuring devices

a) Configure the default gateways according to the following table

Device Name	Interface	IP Address
R0	Gig0/0	192.168.1.1/255.255.255.0
R0	Gig0/1	192.168.2.1/255.255.255.0

(**Note:** make sure to turn on the ports of the router since the default is off)

b) Go to each PC -> Desktop -> IP Configuration and Click on **DHCP**to obtain an IP from the DHCP Service running on the Router

c) Configure DHCP pools on the **DHCP Server** PC as follows:

i. serverPool

Default GW: 192.168.1.1

Start IP: 192.168.1.11 / 255.255.255.0

ii. serverPool2

Default GW: 192.168.2.1

Start IP: 192.168.2.11 / 255.255.255.0

d) **Configure DHCP Relay Agent on the Router**

```
// Configure ip helper-address to forward DHCP discover (broadcasts) from 192.168.2.0/24 network to 192.168.1.2 (DHCP Server address). In here, when router receives the DHCP broadcast, Router becomes and relay agent and sends a uni-cast to DHCP Server (192.168.1.2) on 192.168.1.0/24 network.

R0(config)# int Gig0/1

R0(config-if)# ip helper-address 192.168.1.2 // DHCP Server address
```

```
// Some useful commands to review the configuration

R0# show ip dhcp [pool|binding|conflict]

C:/>ipconfig [/release | /renew]
```

3. Validating the connection

a) Swtich to simulation mode and observe the DHCP handshake (DORA Process – Discover, Offer, Request, Acknowledge)

b) Send an ICMP message from PC1_S0 to PC2_S0

c) Send an ICMP message from PC1_S0 to PC2_S1

LAB 10.5

Task 01 – Simple Local DNS Server

In this task, a simple DNS server is configured to resolve the name **curtin.com** or **www.curtin.com**

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC1_S0
2960	S0
PT-Server	Server

b) Connect them using copper-cables according to the table below:
(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC1_S0	Fa0	S0	Fa0/2
Server	Fa0	S0	Fa0/3

2. Configuring devices

a) PCs need to be given IP addresses (static IP assignment) as shown below:

Device Name	Interface	IP Address	DNS
PC1_S0	Fa0	192.168.1.11/255.255.255.0	192.168.1.2
DNS Server	Fa0	192.168.1.2/255.255.255.0	-

b) Configure DNS on **Server** PC as follows:

i. Enable DNS Service (Services -> DNS and DNS Service: On)

ii. Add the following RRs (Resource Records)

Name	Type	Detail
curtin.com	A Record	192.168.1.2
www.curtin.com	CNAME	curtin.com

c) Enable HTTP Service on **Server** PC (Services -> HTTP: On)

3. Validating the connection

a) Go to PC1_S0

i. Open Command Prompt, type

```
C:/>nslookup curtin.com
C:/>nslookup www.curtin.com
```

ii. Open Web Browser and type **curtin.com** or **www.curtin.com**

b) Swtich to simulation mode and observe DNS resolution process (DNS Query, DNS Answer)

Task 02 – Local DNS Server

In this task, a local DNS server is configured with Multiple RRs (Resource Records). Here we look at SOA (Start of Authority) record along with other RR types.

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC1-Alice
PT-PC	PC1-Bob
2960	S0
PT-Server	DNS Server
PT-Server	Email Server
PT-Server	Web Server

b) Connect them using copper-cables according to the table below:
(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC1-Alice	Fa0	S0	Fa0/4
PC1-Bob	Fa0	S0	Fa0/5
DNS Server	Fa0	S0	Fa0/1
Email Server	Fa0	S0	Fa0/2
Web Server	Fa0	S0	Fa0/3

2. Configuring devices

a) PCs need to be given IP addresses (static IP assignment) as shown below:

Device Name	Interface	IP Address	DNS
PC1-Alice	Fa0	192.168.1.11/255.255.255.0	192.168.1.2
PC1-Bob	Fa0	192.168.1.12/255.255.255.0	192.168.1.2
DNS Server	Fa0	192.168.1.2/255.255.255.0	-
Email Server	Fa0	192.168.1.3/255.255.255.0	-

Web Server	Fa0	192.168.1.4/255.255.255.0	-

b) Configure **Web Server:**

i. Enable HTTP Service

c) Configure **Email Server**

i. Disable HTTP Service

ii. Enable EMAIL Service

iii. Create two user accounts (alice and bob) with the password "cisco"

d) Configure **DNS Server** as follows:

i. Disable HTTP Service

ii. Enable DNS Service

iii. Add the following RRs (Resource Records)

Name	Type	Detail
curtin.com	A Record	192.168.1.4
dns.curtin.com	A Record	192.168.1.2
mail.curtin.com	A Record	192.168.1.3
www.curtin.com	CNAME	curtin.com

iv. Add a SOA Record as below:

// SOA Records indicates which Domain Name Server (DNS) is the best source of information for the specified domain.

Name	Value	Comments
Primary Server Name	dns.curtin.com	Primary name server for the domain
Minimum TTL	100	The default that applies to all of the resource records in the zone
Retry Time	100	The number of seconds before a failed refresh is retried
Mailbox	mail.curtin.com	Email for the domain.
Refresh Time	100	The number of seconds before the zone refreshes
Expiry Time	100	The time, in seconds, before the data is considered unreliable

All "time" values are specified in seconds

4. Validating the connections

a) Go to PC-Alice

i. Open Command Prompt, type

```
C:/>nslookup curtin.com
C:/>nslookup www.curtin.com
```

ii. Open Web Browser and type **curtin.com** or **www.curtin.com**

b) Go to PC-Alice, send an email to bob@curtin.com

c) Go to PC-Bob, receive the email from Alice. Reply to her with a message.

Task 03 – Router as a DNS Client

In this task, a router is configured as a DNS client with a host table (similar to hosts file on a PC).

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC1_R0
2911	R0
PT-Server	Web Server UWA
PT-Server	DNS Server Curtin

b) Connect them using copper-cables according to the table below:

(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC1_R0	Fa0	R0	Gig0/0
Web Server UWA	Fa0	R0	Gig0/1
DNS Server Curtin	Fa0	R0	Gig0/2

2. Configuring devices

a) Devices need to be given IP addresses (static IP assignment) as shown below:

Device Name	Interface	IP Address	DNS
PC1_R0	Fa0	192.168.0.2/255.255.255.0	192.168.0.1
Web Server UWA	Fa0	192.168.1.2/255.255.255.0	192.168.1.1
DNS Server Curtin	Fa0	192.168.2.2/255.255.255.0	192.168.2.1
R0	Gig0/0	192.168.0.1/255.255.255.0	-

R0	Gig0/1	192.168.1.1/255.255.255.0	-
R0	Gig0/2	192.168.2.1/255.255.255.0	-

b) Enable HTTP Service at **Web Server UWA:**

c) Configure **DNS Server Curtin** as follows:

i. Disable HTTP Service

ii. Enable DNS Service

iii. Add the following RRs (Resource Records)

Name	Type	Detail
curtin.com	A Record	192.168.2.2
www.curtin.com	CNAME	curtin.com

d) Configure the router as a DNS Client as follows:

```
R0(config)# ip domain-lookup // by default domain lookup is enabled. This is why if you enter a uknown command (e.g. “reload”), the router will try to resolve it via DNS name resolution. You can disable this by “no ip domain-lookup” command”

R0(config)# ip name-server 192.168.2.2 // Router will forward the DNS queries to the name server specified if not found on the hosts table configured below

R0(config)# ip host uwa.com 192.168.1.2 // Adding entries to the host table on the router.
```

5. Validating the connection

a) Go to Router (**R0**)’s CLI and switch to simulation mode

i. ping uwa.com

ii. ping curtin.com

LAB 10.6

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	Birdy
PT-PC	Jerry
PT-PC	Tom
2960	S1

2960	S2
2911	R0
PT-Server	Email-Web-DNS-Server-01
PT-Server	Email-Web-DNS-Server-02

b) Connect them using copper-cables according to the table below:
(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
R0	Gig0/1	S1	Fa0/1
Email-Web-DNS-Server-01	Fa0	S1	Fa0/2
Jerry	Fa0	S1	Fa0/3
Tom	Fa0	S1	Fa0/4
R0	Gig0/2	S2	Fa0/2
Email-Web-DNS-Server-01	Fa0	S2	Fa0/3
Birdy	Fa0	S2	Fa0/4

2. Configuring devices

a) Devices need to be given IP addresses (static IP assignment) as shown below:

Device Name	Interface	IP Address	Default GW	DNS
R0	Gig0/1	192.168.1.1/255.255.255.0	-	-
Email-Web-DNS-Server-01	Fa0	192.168.1.2/255.255.255.0	192.168.1.1	-
Jerry	Fa0	192.168.1.3/255.255.255.0	192.168.1.1	192.168.1.2
Tom	Fa0	192.168.1.4/255.255.255.0	192.168.1.1	192.168.1.2
R0	Gig0/2	192.168.2.1/255.255.255.0	-	-
Email-Web-DNS-Server-02	Fa0	192.168.2.2/255.255.255.0	192.168.2.1	-
Birdy	Fa0	192.168.2.3/255.255.255.0	192.168.2.1	192.168.2.2

b) Disable HTTP Service on **Email-Web-DNS-Server-01**, **Email-Web-DNS-Server-02**.

c) Enable EMAIL Service on **Email-Web-DNS-Server-01**, **Email-Web-DNS-Server-02**.

d) Configure DNS Servers as follows:

i. Enable DNS Service on all DNS Servers

ii. Add the following RRs (Resource Records)

DNS Server	Name	Type	Detail
	mail.catandmouse.com	A Record	192.168.1.2

Email-Web-DNS-Server-01	acme.com	A Record	192.168.2.2
Email-Web-DNS-Server-02	mail.acme.com	A Record	192.168.2.2
	catandmouse.com	A Record	192.168.1.2

e) Create the following email accounts

i. tom@catandmouse.com, jerry@catandmouse.com at **Email-Web-DNS-Server-01**

ii. birdy@acme.com at **Email-Web-DNS-Server-02**

3. Validating the connection

a) Send an email from Tom's PC to Jerry's PC

b) Send an email from Tom's PC to Bird's PC

c) Reply to the email from Bird's PC to Tom's PC

d) Observe all communications in the simulation mode

LAB 10.7

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC-Alice
PT-PC	PC-Bob
2960	S1
2960	S2
2911	R0
PT-Server	DNS Server Curtin
PT-Server	Email Server Curtin
PT-Server	DNS Server Pub
PT-Server	DNS Server UWA
PT-Server	Web Server UWA
PT-Server	Email Server UWA

b) Connect them using copper-cables according to the table below:
(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC-Alice	Fa0	S2	Fa0/1

DNS Server Curtin	Fa0	S1	Fa0/2
Email Server Curtin	Fa0	S1	Fa0/3
R0	Gig0/0	S1	Fa0/4
DNS Server Pub	Fa0	R0	Gig0/2
R0	Gig0/1	S2	Fa0/4
DNS Server UWA	Fa0	S2	Fa0/5
Web Server UWA	Fa0	S2	Fa0/6
Email Server UWA	Fa0	S0	Fa0/7
PC-Bob	Fa0	S2	Fa0/8

2. Configuring devices

a) Devices need to be given IP addresses (static IP assignment) as shown below:

Device Name	Interface	IP Address	Default GW	DNS
PC-Alice	Fa0	192.168.1.11/255.255.255.0	192.168.1.1	192.168.1.3
DNS Server Curtin	Fa0	192.168.1.3/255.255.255.0	192.168.1.1	-
Email Server Curtin	Fa0	192.168.1.4/255.255.255.0	192.168.1.1	-
DNS Server Pub	Fa0	192.168.3.2/255.255.255.0	192.168.3.1	-
DNS Server UWA	Fa0	192.168.2.5/255.255.255.0	192.168.2.1	-
Web Server UWA	Fa0	192.168.2.6/255.255.255.0	192.168.2.1	-
Email Server UWA	Fa0	192.168.2.7/255.255.255.0	192.168.2.1	-
PC-Bob	Fa0	192.168.2.12/255.255.255.0	192.168.2.1	192.168.2.5
R0	Gig0/0	192.168.1.1	-	-
R0	Gig0/1	192.168.2.1	-	-
R0	Gig0/2	192.168.3.1	-	-

b) Disable HTTP Service on **DNS Server Curtin**, **DNS Server Pub**, **DNS Server UWA** and **Email Server UWA, Email Server Curtin.**

c) Enable HTTP Service on **Web Server UWA** and Email Service on **Email Server UWA** and **Email Server Curtin**

d) Configure DNS Servers as follows:

i. Enable DNS Service on all DNS Servers

ii. Add the following RRs (Resource Records)

DNS Server	Name	Type	Detail
DNS Server Curtin	com	NS	root
	root	A Record	192.168.1.2
	mail.curtin.com	A Record	192.168.1.4
DNS Server Pub	ns1.uwa.com	A Record	192.168.1.5
	uwa.com	SOA	ServerName: uwa.com Mailbox: uwa.com Expiry: 30 Refresh: 20 Retry: 10 MinTTL: 5
	uwa.com	NS	ns1.uwa.com
DNS Server UWA	dns.uwa.com	A Record	192.168.1.5
	uwa.com	A Record	192.168.1.6
	uwa.com	SOA	ServerName: dns.uwa.com Mailbox: mail.uwa.com Expiry: 45 Refresh: 30 Retry: 10 MinTTL: 15
	mail.uwa.com	A Record	192.168.1.7

Note: **DNS Server Pub** only has NS records including the A records for the name servers.

// How does SOA Records Work?

// Assume the domain name **uwa.com** needs to be resolved from **PC-Alice**

- The DNS resolver of the PC will send a query to **DNS Server Curtin**
- **DNS Server Curtin** checks whether it has an A Record for uwa.com (Authoritative Answer).
    - If not found, it will check in its DNS cache (Non-authoritative answer)
    - If not found, **DNS Server Curtin** will forward to query to **DNS Server Pub** (TLD - Top Level Domain Server)
- **DNS Server Pub** will repeat the same steps as DNS Server Curtin and if failed, it will forward the query to **DNS Server UWA**
- **DNS Server UWA** will have an A record (Authoritative Answer) for uwa.com
    - In addition, if a **SOA (Start of Authority)** record is defined on the **uwa.com**, the parameters of the SOA record for uwa.com will be applied to the DNS query sender **(DNS Server Pub).**

For e.g. is the SOA for uwa.com says the TTL is 5 seconds, **DNS Server Pub** will only keep the A record for **uwa.com** and **mail.uwa.com** (mailbox) for 5 seconds in **DNS Server Pub's DNS cache**. The SOA from **DNS Server UWA** will not be applied to **DNS Server Curtin**

- If a SOA record exists for the **NS** (nameserver) **record** of **uwa.com** at **DNS Server Pub,** its parameters will be applied on **DNS Server Curtin** (since it is the DNS query sender for **DNS Server Pub**).

If a SOA record does not exist for a domain name, the non-authoritative answer received by the querying server will stay in its DNS cache indefinitely (DNS cache needs to be manually cleaned).

e) Create two email accounts (alice@curtin.com and bob@uwa.com) at **DNS Server Curtin** and **DNS Server UWA** respectively.

i. Also, configure the mailboxes at **PC-Alice** and **PC-Bob**accordingly.

3. Validating the connection

a) Validate the following test cases

Source Device	Test Case	Comment
PC-Alice	nslookup for uwa.com	Observe the DNS request forwarding in simulation mode. How long the fetched record will be kept and DNS cache of DNS Server Pub and DNS Server Curtin respectively? (Hint: look at the SOA record for the TTL)
PC-Alice	Open the web browser and type uwa.com	Must display a page (index.htmk) from **Web Server UWA**
PC-Alice	Send an email to bob@uwa.com; See whether Bob has received the email **(Fail)**	This is because, when the DNS query for uwa.com from Alice's mail server (**Email Server Curtin**) is resolved by **DNS Server UWA**, it resolves the name uwa.com to **Web Server UWA**. But in real world, the MTA (Mail Transfer Agent) – **Email Server Curtin** looks for MX record first, and the A record next at **DNS Server UWA**. Packet Tracer does not support MX records. If you create an email account for bob@uwa.com at **Web Server UWA**, you will discover the email sent by Alice in this case. This is analogous to the free email account attached to the web domain you would see in real world (for

		e.g. web domain: aol.com, email: user@aol.com on the same web server)
PC-Alice	Send an email to alice@curtin.com; See whether Bob has received the email **(Fail)**	We have only configured proper DNS entries for Alice to reach Bob or uwa.com domain. But not vice versa. You would have to configure the proper DNS entries on **DNS Server UWA**, **DNS Server Pub** and **DNS Server Curtin** to get this to work.
PC-Alice	Open the web browser and type curtin.com **(Fail)**	Same as above

LAB 10.8

1. Setting up the logical view of the network

a) Insert the following devices onto the logical workspace and name them accordingly

Device Type	Device Name
PT-PC	PC0
PT-PC	PC1
PT-PC	PC2

b) Connect them using copper-cables according to the table below:
(**Note:** use the correct cable configuration. i.e. cross-over or straight through)

Device Name	Interface	Device Name	Interface
PC0	Fa0	R0	Gig0/2
PC1	Fa0	R1	Gig0/1
PC2	Fa0	R2	Gig0/2

2. Configuring devices

a) Devices need to be given IP addresses (static IP assignment) as shown below:

Device Name	Interface	IP Address	Default GW
PC0	Fa0	192.168.0.2/255.255.255.0	192.168.0.1
PC1	Fa0	192.168.1.2/255.255.255.0	192.168.1.1
PC2	Fa0	192.168.2.2/255.255.255.0	192.168.2.1
R0	Gig0/2	192.168.0.1/255.255.255.0	-
R1	Gig0/1	192.168.1.1/255.255.255.0	-
R2	Gig0/2	192.168.2.1/255.255.255.0	-

b) Configure UDP Sockets on each of the PCs:

i. Go to Programming -> New, template **UDP Server - Python**

ii. Name the projects in PC0, PC1, PC2 to UDP0, UDP1, UDP2 respectively

c) Add/Change the UDP Server – Python project code as shown below:
Change the line "`socket.send(…)`" to:

i. On **PC0:**

```
socket.send("192.168.0.1", 1235, "hello " + str(count) + " from PC0")

socket.send("192.168.0.2", 1235, "hello " + str(count) + " from PC0")
```

ii. On **PC1:**

```
socket.send("192.168.0.0", 1235, "hello " + str(count) + " from PC1")

socket.send("192.168.0.2", 1235, "hello " + str(count) + " from PC1")
```

iii. On **PC2:**

```
socket.send("192.168.0.0", 1235, "hello " + str(count) + " from PC2")

socket.send("192.168.0.1", 1235, "hello " + str(count) + " from PC2")
```

d) Now **Run** all the projects on **PC0**, **PC1**, **PC2**

3. Validating the connection

a) Check whether UDP data is received by PC0 from PC1, PC2
b) Check whether UDP data is received by PC1 from PC0, PC2
c) Check whether UDP data is received by PC2 from PC0, PC1

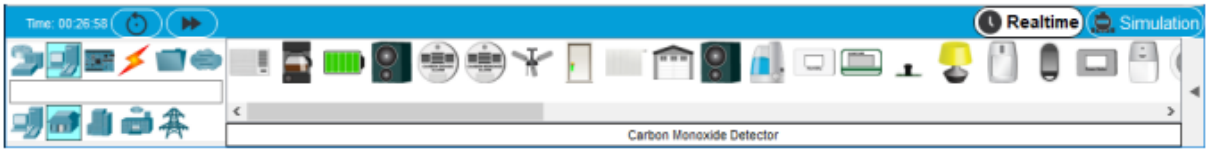
CNCO2000 – Practical 11

P11: Networking with IoT

Q1:Understand the Basics of IoT Where are the Things? In addition to routers and switches, the Network Component Box now contains a variety of Smart Things and components. They can be created by selecting the device and then clicking on a blank area either on the Logical Workspace or the Physical Workspace.

Smart Things are physical objects that can connect to the Registration Server or Home Gateway through a network interface. They are separated into subcategories under End Devices. The

categories consist of Home, Smart City, Industrial, and Power Grid. Pictured below are a few Smart Thing devices that are in the Home category.



Components are physical objects that connect to microcontroller (MCU-PT) or single boarded computers (SBC-PT) and typically does not have a network interface. These are simple devices that only communicate through their analog or digital slots.

There are three subcategories for Components, they are:

• Boards: microcontrollers (MCU-PT), single boarded computers (SBC-PT), and a special device called Thing. Things are used to create self-contained physical objects like coffee makers or smoke alarms.

• Actuators: these components manipulate the Environment, themselves, or the area around them.

• Sensors: these components sense the Environment (photo detectors, temperature sensor), the area around them (RFID, metal sensor), or interactions (potentiometer, push button).

Below is an image some sensor components:

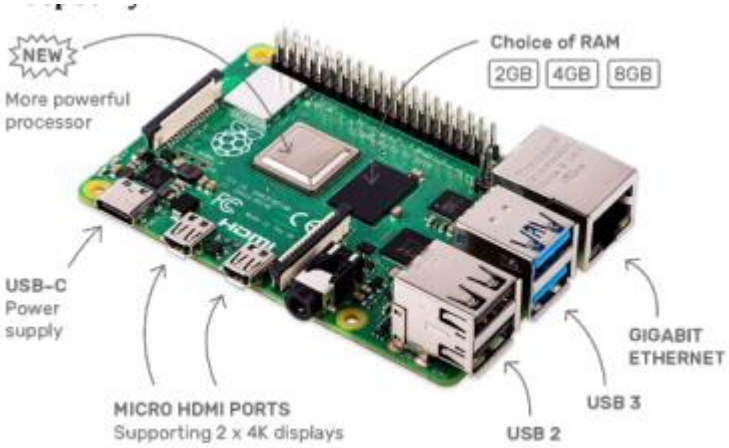
Examples for MCU (Microcontroller Unit) in real world

Arduino

(Note: There is NO CPU – central processing unit but an IC (Integrated Circuit) component on the board. For heavy CPU or GPU intensive work, use a SBC.)

Examples for SBC (Single Board Computers) in real world

Raspberry Pi 4



Nvidia Jetson (CPU + GPU for AI (Artificial Intelligence) tasks)

Interacting with Smart Things

There are many ways to interact with Things. The Specifications tab of the Device Dialog will list the types of interactions that are allowed for that specific Thing.

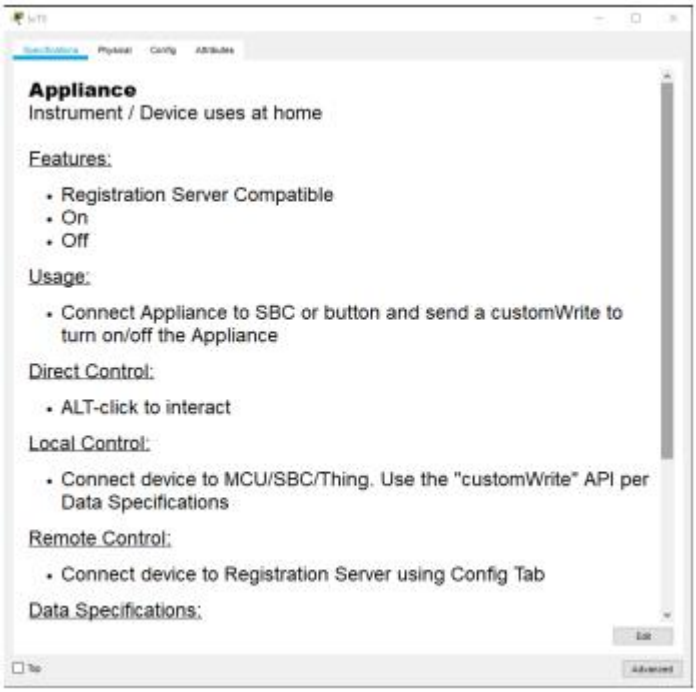
The types of interactions are:

• Direct Control: interact with the object directly on the workspace using the mouse. The mouse interactions are usually modified by pressing "ALT" on the keyboard at the same time as moving the mouse or clicking.

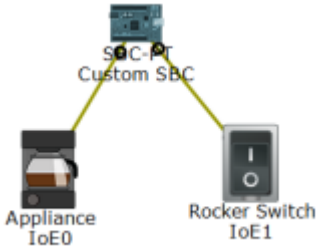
• Local Control: interact with the object using the digital or analog slots.

• Remote Control: interact with the object over an IP network. In most cases, this means interacting with the Registration Server or Home Gateway.

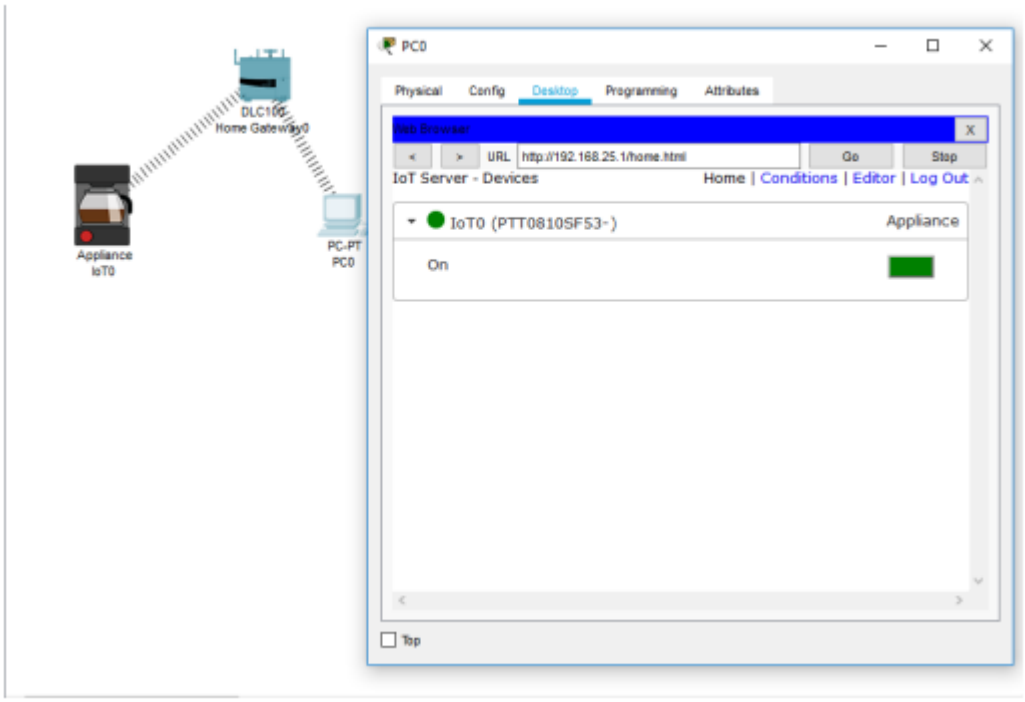
Below is a sample specification for the prepackaged Thing, Appliance. In this case, ALT-clicking directly on the Appliance in the workspace will turn it on.



It is also possible to control the Appliance locally through the digital and analog ports as shown in the image below. The SBC sends an on or off command to the Appliance depending on the Rocker Switch state.

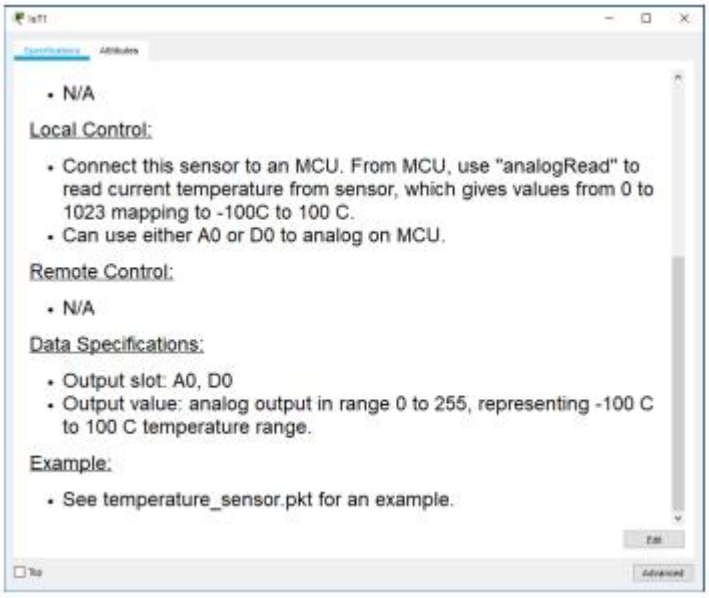


Finally, it is possible to remotely control the Thing through the Registration Server or Home Gateway. The Appliance is registered to the DLC100 Home Gateway. The Home Gateway also provides wireless connectivity to the PC and Appliance. Using the PC and connecting to the Home Gateway web service, it is possible to remotely control the Appliance over the network.



Interacting with Components

Components do not have network interface cards. Typically, they are connected to the digital or analog slots on the MCU or SBC. It is important to read the Specifications page for the component because it typically describes the data specifications. For example, see the specification for the Temperature Sensor below:



Cabling Things

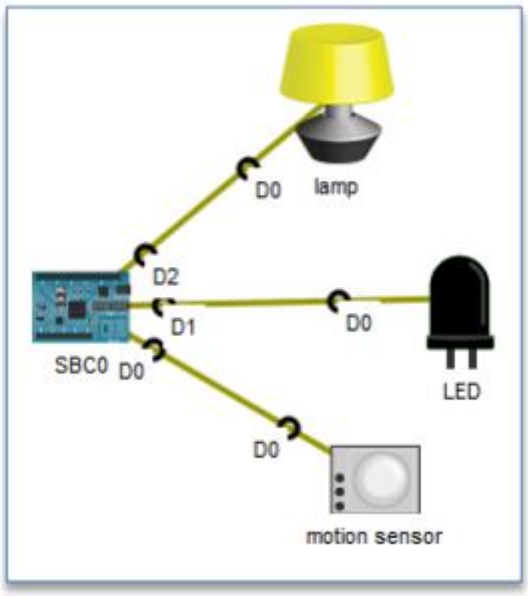
Cabling Things is similar to cabling networking devices. Please refer to the Workspace Basics, Logical Workspace for more information. The main difference is the addition of digital and analog slots (D0, D1, A0, A1, etc). These slots only accept the IoE Custom Cable. The IoE Custom Cable is a bundle of

wires, which are inaccessible to the user, that provides all the ground, power, and data connections necessary for communication between all Things, components, MCU, and SBCs.

Currently, the model does not let you unwrap the IoE Custom cable and create circuit schematics.

Page Q2: Configuring a simple network of things

• On a new .pkt file, implement the simple IoT network shown below:



Please connect the devices to the correct pins (DO, D1, D2) of SBC0 as shown on the diagram above.

a. Devices

Device Type	Device Name
SBC Board	SBC0
LED	LED
Light	lamp
Motion Sensor	motion sensor

Use PT Official Help to find the location of the IoT devices on Packet Tracer device bar.

To read the specification of each device (API to use, device information, states, etc.) click on the Device.

b. Go to SBC->Programming

c. Create a project named “Motion-Python”

d. Write the code below:

```
from gpio import *
from time import *

motion = 0

# Read from a the motion sensor
def readFromSensor():
      global motion
      motion = digitalRead(0) # Read from slot 0 using digitalRead API
      print "readFromSensor(): motion " + str(motion)

def writeToDevices():
      global motion
      print "writeToDevices(): motion " + str(motion)

      if motion == 1023:
            print "HIGH"
            # Write the lamp state "2" (2=ON, 1=Dim, 0=Off) on slot 2 using
            customWrite API. Click on the lamp to see the details of the states
            customWrite(2, 2)

            # Write the LED state "1023" or HIGH (1023=HIGH, 0=Off) on slot 1
            using analogWrite API
            analogWrite(1, HIGH)
      else:
            print "LOW"
            customWrite(2, 0)
            analogWrite(1, LOW)

def main():
      # Initialize the connected pins of SBC
      pinMode(0, IN)
      pinMode(1, OUT)
      pinMode(2, OUT)

      while True:
            readFromSensor()
            writeToDevices()
            delay(5000) # wait for some time (i.e. 5 sec) before reading motion
                        sensor input

if __name__ == "__main__":
      main()
```


from gpio import *

from time import *

motion = 0

Read from a motion sensor

def readFromSensor():

  global motion

  motion = digitalRead(0) # Read from slot 0 using digitalRead API

  print "readFromSensor(): motion " + str(motion)

```
def writeToDevices():
    global motion
    print "writeToDevices(): motion " + str(motion)

    if motion == 1023:
        print "HIGH"
        # Write the Lamp state "2" (2=ON, 1=Dim, 0=Off) on slot 2 using
        # customWrite API. Click on the Lamp to see the details of the states
        customWrite(2, 2)

        # Write the LED state "1023" or HIGH (1023=HIGH, 0=Off) on slot 1
        # using analogWrite API
        analogWrite(1, HIGH)
    else:
        print "LOW"
        customWrite(2, 0)
        analogWrite(1, LOW)

def main():
    # Initialize the connected pins of SBC
    pinMode(0, IN)
    pinMode(1, OUT)
    pinMode(2, OUT)

    while True:
        readFromSensor()
        writeToDevices()
        delay(5000)  # wait for some time (i.e. 5 sec) before reading motion sensor input

if __name__ == "__main__":
```

main()

e. Run the project

f. Press Alt key and move the mouse on top of the motion sensor, wait for 5 seconds

i. Lamp and LED must be turned on

ii. Wait another 5 seconds without moving the mouse over the sensor

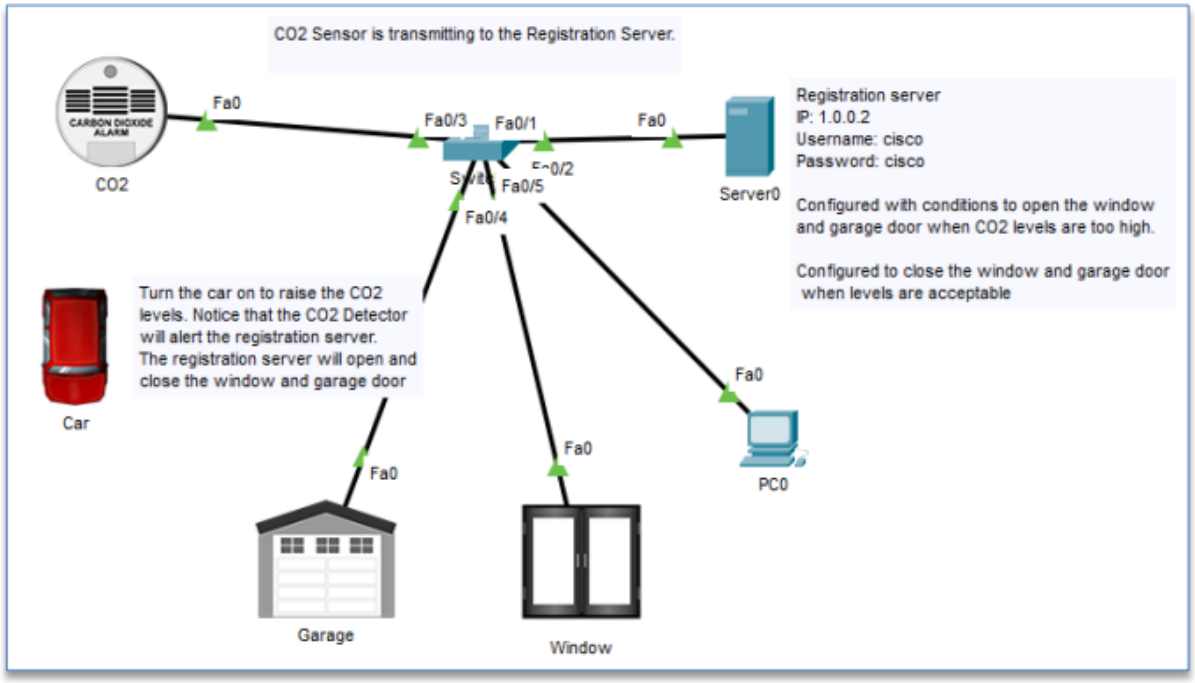
iii. Lamp and LED must be turned off.

iv. Observe the console prints of the program to better understand code

g. Swtich to simulation mode and observe signal transmission.

Q3: Configuring a IoE (Internet of Everything) Registration Server

• On a new .pkt file, implement the IoT network shown below:



a. In order to communicate with a Registration Server, the devices have to be configured with an IP address. Use the following table to assign the IP addresses accordingly

Device type	**Device Name**	**IP**	**Gateway (optional)**
PT-Server	Server0	1.0.0.2/255.0.0.0	1.0.0.1
Carbon Dioxide Detector	CO2	1.0.0.3/255.0.0.0	1.0.0.1
Garage Door	Garage	1.0.0.4/255.0.0.0	1.0.0.1
Window	Window	1.0.0.5/255.0.0.0	1.0.0.1
PT-PC	PC0	1.0.0.5/255.0.0.0	1.0.0.1

b. Go to Server0 (Registration Server) and enable IoT Service (Services -> IoT Service)

c. Open the web browser on PC0 and type 10.0.0.2 to access the Server0 (IoT Registration Server) to create a new account on Server0

i. username: cisco, password: cisco

ii. Once the account is created, Go to Server0 -> Services -> IoT, see whether the newly created account is shown there with the credentials.

d. Devices must be configured to communicate with the Server0 (IoT Registration Server) on the network as below:

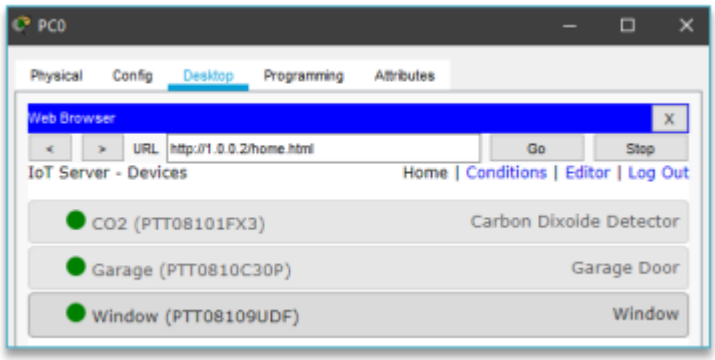
Device Type	**Device Name**	**Settings -> IOT Server**
Carbon Dioxide Detector	CO2	Enable Remote Server Server Address: 1.0.0.2 Username: cisco Password: cisco
Garage Door	Garage	Enable Remote Server Server Address: 1.0.0.2 Username: cisco Password: cisco
Window	Window	Enable Remote Server Server Address: 1.0.0.2 Username: cisco Password: cisco

e. Lets configure the logical conditions to open the garage door and window once a hazardous level of CO2 is detected.

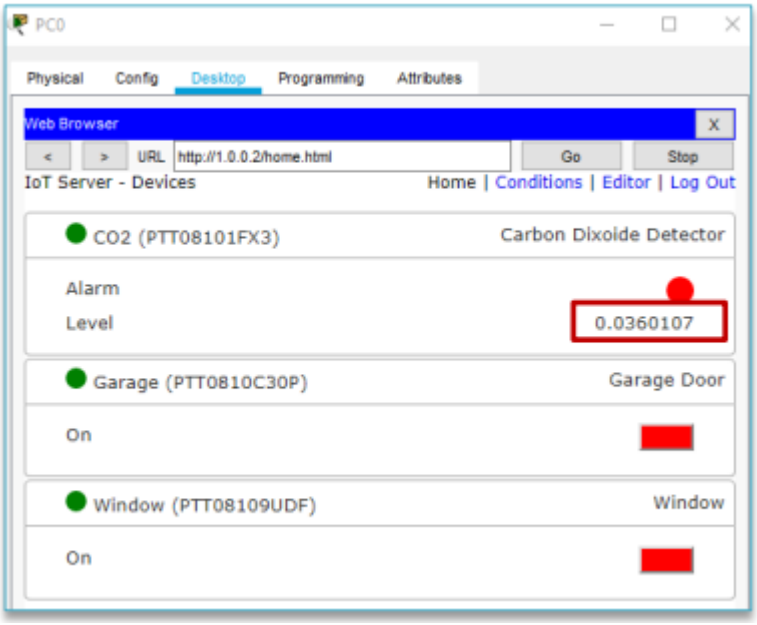
i. Access the web portal of the server from PC0 (via the web browser)

Click on the link Home

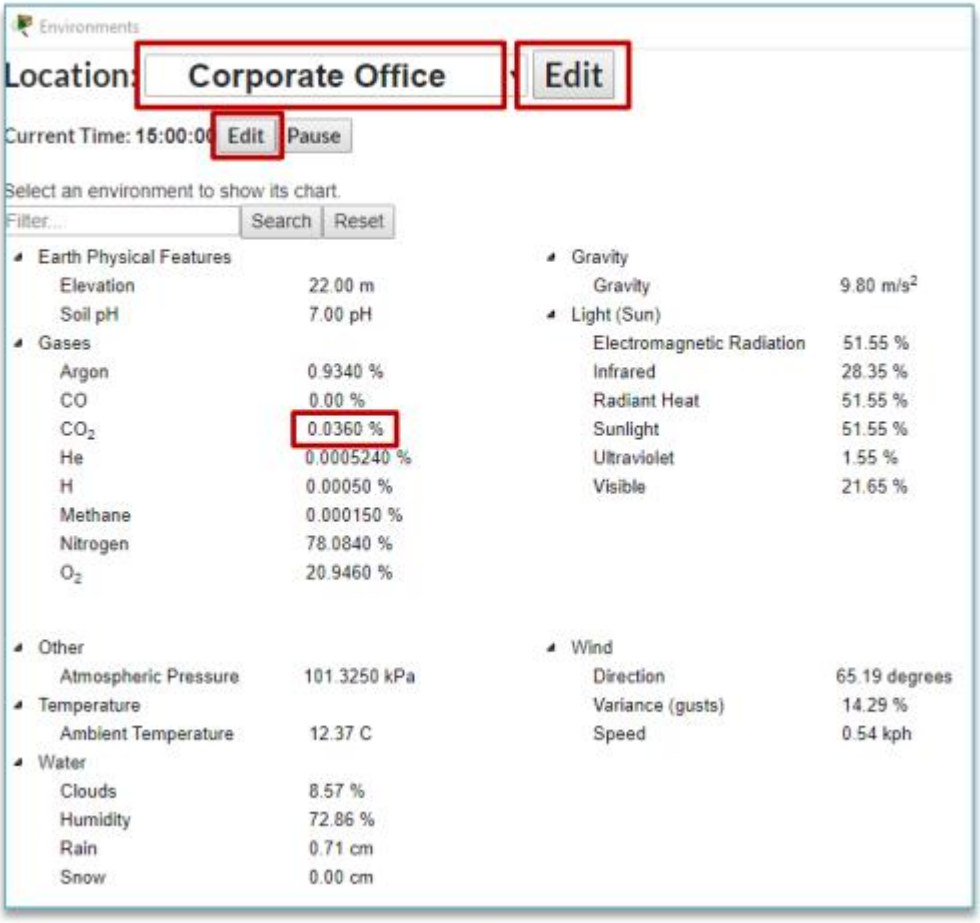
The IoT devices must be shown as below: (Click on the device name below to show the status)



Click on the device name to show the status (current CO2 reading, etc.)

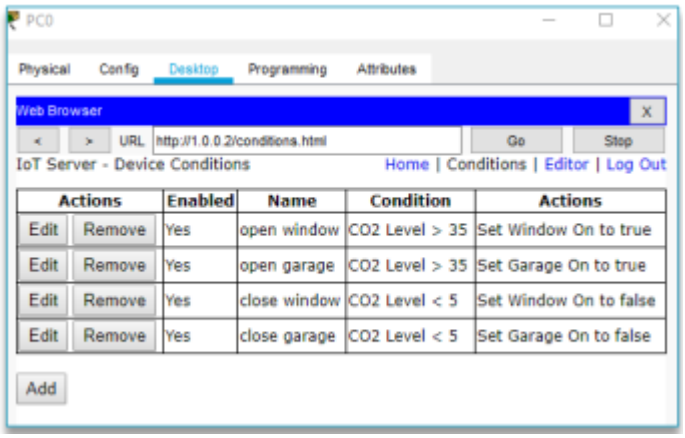


Validate the CO2 level of the container where all these devices are placed. Click on Environments button: (Physical Layer Container: Cooperate Office)

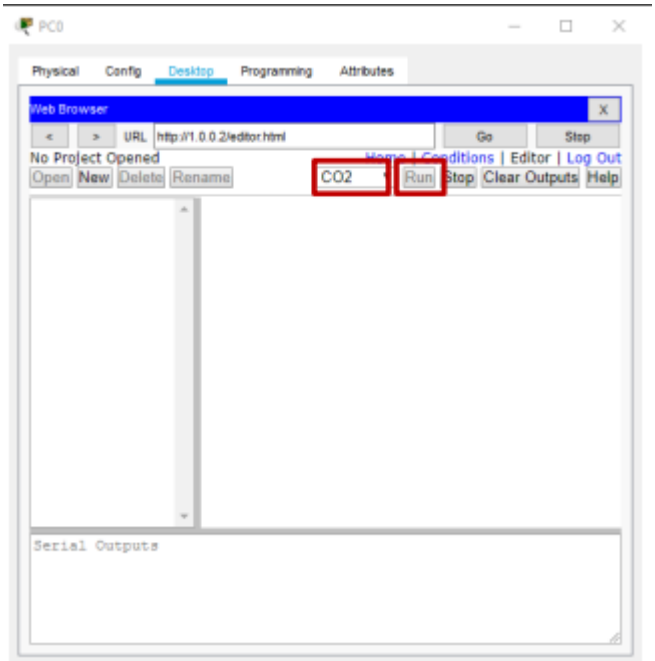


Note: You can click on Edit (next to location) to simulate different environment conditions with key points. In PT there IoT Devices that will detect these environmental changes (CO2, CO, H, O2, Visible Light, etc.).

ii. Click on the link Conditions and add following conditions

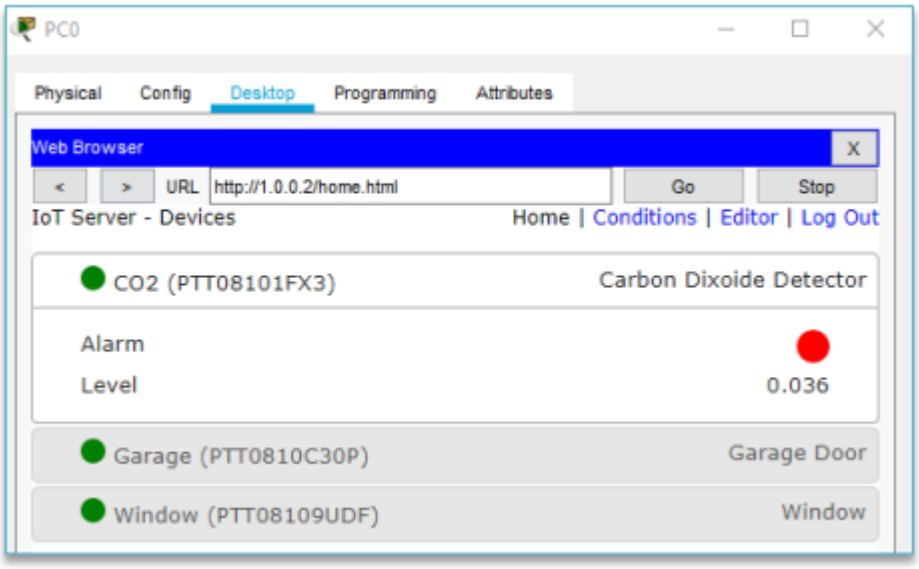


iii. Optional – Click on the link Editor. This will enable you to write programs on individual devices to work with custom operations defined on them. For this activity, we skip this part.



Note: Make sure the devices are in Run state (Active state)

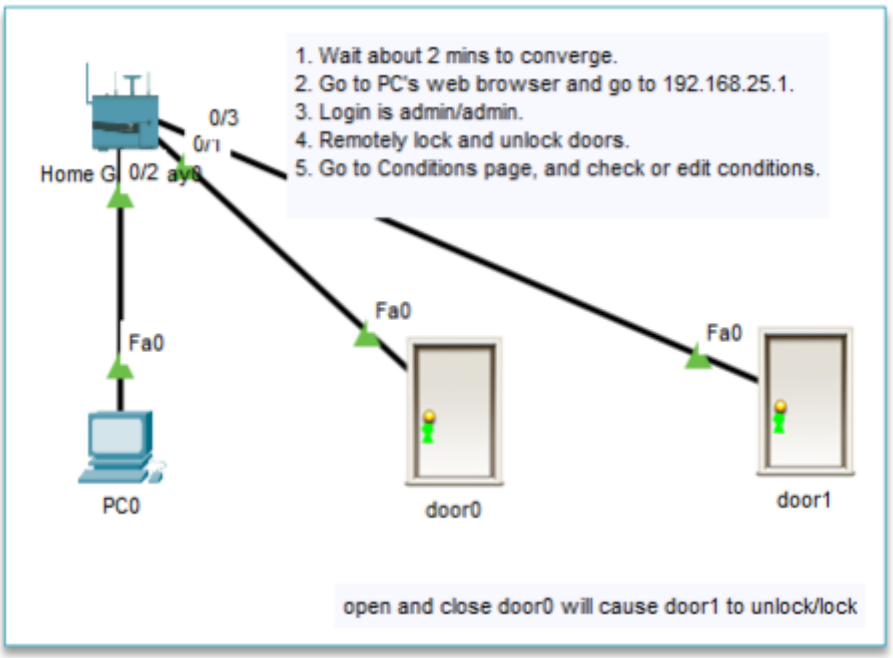
iv. Go back to Home and expand the CO2 Sensor detail tab, keep it open where visible before you do the next step. Watch closely on CO2 Level. (You can also view the CO2 level of current physical container by going to Environments windows shown above)



f. Press Alt and click on the Car. Observe what happens next!

Q4: Configuring a Home Gateway

• On a new .pkt file, implement the IoT network shown below:



Note: Home Gateway is like a Router that connect the local network to the internet (Ports: 4 LAN ports, 1 internet port). But it has IoT Registration Server in built (go to GUI tab and enable HTTP) and can be accessed via a web browser of a PC.

a. Devices

Device Type Device Name

Home Gateway Home Gateway0

Door door0

Door door1

PT-PC PC0

b. Configure gateway address on Home Gateway

i. Go to Home Gateway -> Config -> LAN

ii. IP: 192.168.25.1 / 255.255.255.0

c. In order to communicate with the Home Gateway, the devices have to be configured with an IP address. In PT (Packet Tracer), the Home Gateway Device is running a DHCP Service by default. Therefore, we can configure the other devices to use DHCP to obtain an IP automatically. Also, configure the devices to use the Home Gateway as the IoT Server

d.

Device Name FastEthernet0 - IP Config->Settings

door0 via DHCP Enable Home Gateway

door1 via DHCP Enable Home Gateway

PC0 via DHCP Enable Home Gateway

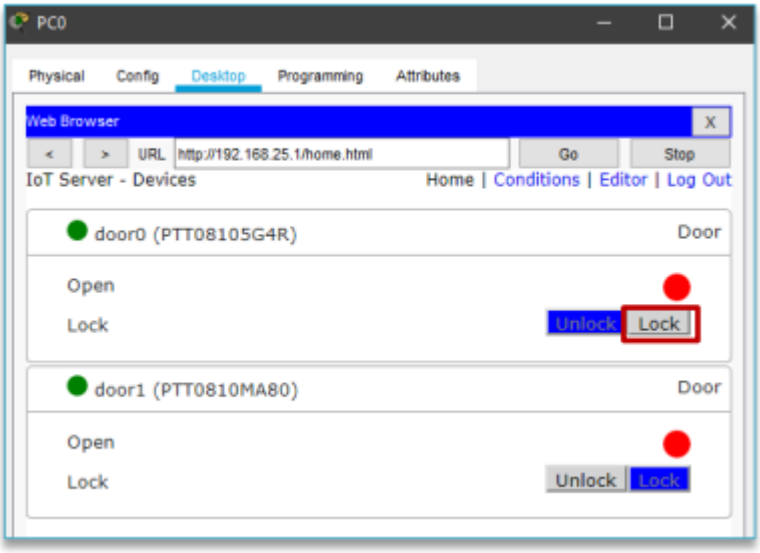
Note: Unlike using a separate remote server as a registration server that needs credentials (extra layer of security) to communicate, using a Home Gateway does not require the credentials. This is analogous to a regular device (i.e. PC) having a wired connection to a router does not require any special credentials to work with the router. In contrast, a wireless connection would require a password for a device to be connected to an broadcasted SSID.

e. In order to configure the rules for the IoT devices connected to the Home Gateway, open a web browser on PC0 and go to 192.168.25.1 (login with the credentials UN: admin, PW: admin / Default account)

f. Configure the rules as shown below: g. Everything is configured! Let's see it in action …

h. Open and close door0 will cause door1 to unlock/lock (press Alt + door0 to interact with the door)

i. You can also lock unlock door0 via the web portal remotely:



Q5: Try me! Questions

1. In Q2, what is the importance of the use of “wait()” function inside the main while loop? Is it really necessary? What could be a better way to replacing expectation of using the “wait()” function?

2. In Q3, why it is not possible to unlock, lock door1 via the web portal remotely? 3. Open Packet Tracer -> File -> Open Samples, Try different IoT implementation samples in:

a. IoT/Environment

b. IoT/iot

c. IoT/IoT Devices

d. IoT/Programming/bluetooth music player

e. IoT/Programming/desktop user gui desktop user gui is an example of deploying a program code into a Desktop App with a GUI which could be installed on a PC/SBC device. (Sample GUI Project: PC -> Programing -> Desktop App - Python)